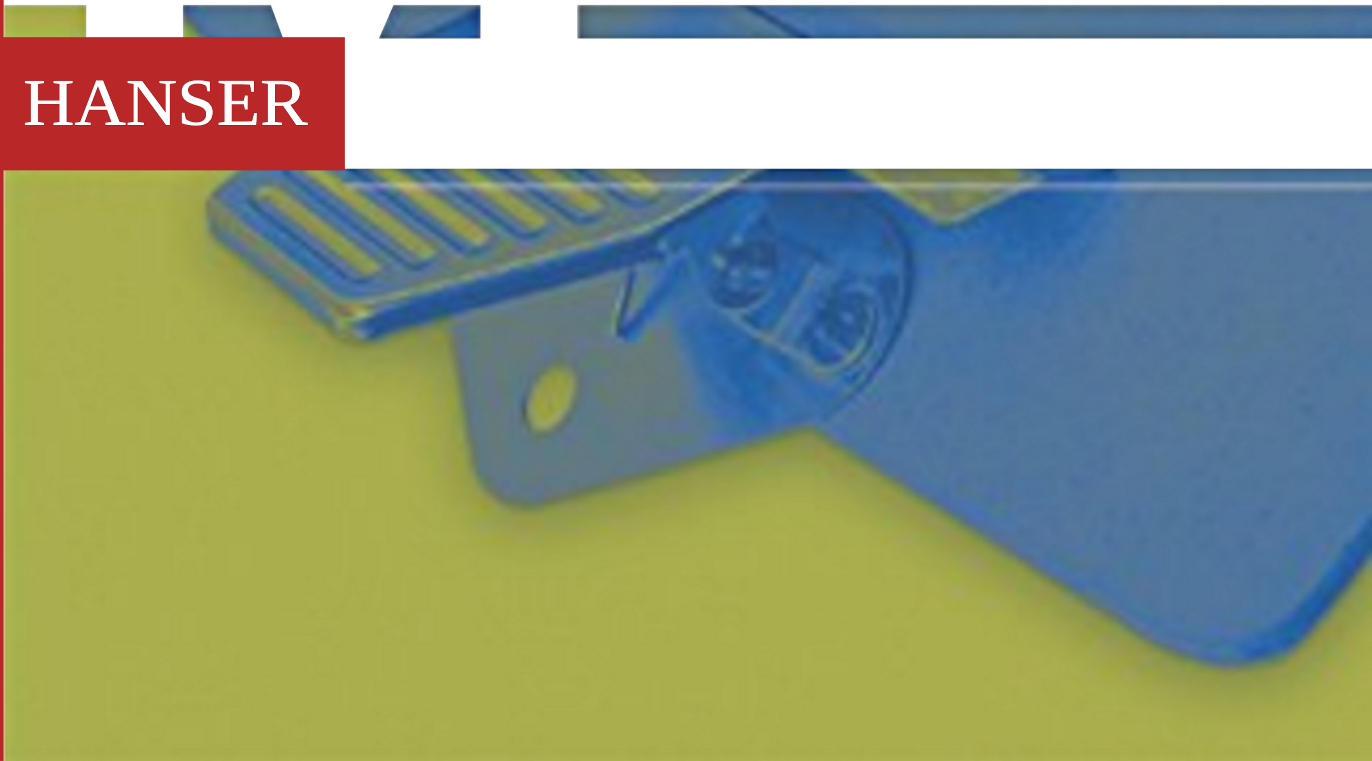
René Martin

OpenOffice-Programmierung

Automatisieren von Büroanwendungen mit Basic

HANSER



HANSER

OpenOffice-Programmierung

René Martin

Automatisieren von Büroanwendungen mit Basic

ISBN 3-446-40450-3

Vorwort

Weitere Informationen oder Bestellungen unter
http://www.hanser.de/3-446-40450-3 sowie im Buchhandel

Vorwort

Zu OpenOffice.org/StarOffice

Eine spannende Sache ist es schon: Eine OpenSource-Gemeinde, die sich das hohe Ziel setzt, ein Anwendungsprogramm kostenfrei als OpenSource-Produkt allen Benutzern zur Verfügung zu stellen, und damit sogar noch den Anspruch erhebt, Microsoft Konkurrenz machen zu wollen. Immerhin: Download-Zahlen, Usergroups und Artikel im Internet sprechen ein deutliche Sprache. Mehr und mehr Anwender arbeiten und beschäftigen sich mit diesem Office-Paket. OpenOffice.org stellt alles zur Verfügung, was von einem Office-Programm erwartet wird. Man erstellt Dokumente, kalkuliert, analysiert Daten, entwirft Präsentationen und öffnet Datenbanken. StarOffice/OpenOffice.org kann Dokumente in das Portable Document Format (.pdf) exportieren und Grafiken im Flash-Format (*.swf) weiterreichen – ohne auf externe Software ausweichen zu müssen. Internationale Nutzer wurden durch die Integration von Complex Text Layout-Sprachen (CTL) (wie beispielsweise Thai, Hindi, Arabisch und Hebräisch) sowie vertikale Schreibrichtungen berücksichtigt. Doch all dies wäre sicherlich nur ein Teil, wenn man das Paket nicht auch von innen und außen „steuern“ könnte.

Zu diesem Buch

Und eben um eine dieser Programmiersprachen soll es in diesem Buch gehen. Seit einigen Monaten häufen sich die Berichte über OpenOffice.org und StarBasic *. Nicht nur in Fachzeitschriften erscheinen mehr und mehr Artikel zu diesen kostenlosen, beziehungsweise kostengünstigen Office-Produkten, auch die Anzahl der Bücher zu OOo und StarOffice

* Im Folgenden werde ich der Einfachheit wegen von OOo sprechen – ich meine dabei selbstverständlich beide Produkte, StarOffice und OpenOffice.org.

nimmt zu. Bis zur Drucklegung dieses Buches lag allerdings noch kein deutschsprachiges Buch zum Thema des Automatisierens vor **. Diese Lücke möchte ich hiermit füllen.

Parallel dazu entstehen mehr und mehr Internet-Communities, es existieren einige pdf-Dokumente und Seiten zur Programmierung von OOo im Internet. Die wichtigsten von ihnen sind am Ende des Vorworts aufgelistet.

Zum Autor

Seit über 15 Jahren unterrichte ich Softwareprodukte und Programmiersprachen von verschiedenen Herstellern aus verschiedenen Bereichen. Dabei zählen die Basic-Dialekte (oder sollte ich besser sagen: Basic-Sprachen?) zu meinen Lieblingssprachen. Vielleicht weil ich in meiner Tätigkeit als Selbstständiger eine Lücke gefunden habe: das Einrichten der Standardapplikationen zum Erstellen von Texten, zum Kalkulieren von Daten, Präsentationssoftware, Datenbanken und so weiter. Diese Programme stellen eine Reihe von Vorlagen, Assistenten, Hilfen und Werkzeugen zur Verfügung. Aber es liegt in der Natur der Sache, dass sie nicht alles können. Oder bestimmte Rechtestrukturen oder Arbeitsabläufe sollen sich für den Benutzer transparent in den Programmen wieder finden. Was die Standardsoftware OOo nicht von Hause aus kann, das kann man programmieren, mit Basic, und seit der Version 2.0 auch mit Java und Python.

Und hier setzt der zweite Teil meiner Begeisterung an: Ich bin immer wieder überrascht und erstaunt, wie viel ich programmieren kann, das heißt, wie viele Benutzereingaben ich überprüfen kann, wie viele Systemvoraussetzungen ich kontrollieren kann und wie tief ich mit Basic im System komme.

Diese Begeisterung versuche ich seit Jahren in den vielen Büchern weiterzugeben, die ich bereits über StarCalc und VBA (und verwandte Themen) publiziert habe; und in den vielen Artikeln in Fachzeitschriften, für die ich seit Jahren schreiben darf.

Zu den Lesern des Buchs

Ich habe versucht, das vorliegende Buch für Anfänger, Fortgeschrittene und Profis zu schreiben. Dabei bin ich mir im Klaren, dass der Begriff „Profi“ sehr weit und sehr unterschiedlich interpretiert werden kann. Dieser Gefahr bin ich mir bewusst; das Ergebnis sehe ich im Schulungsbereich in „Aufbaukursen“. Deshalb habe ich versucht, jedes Kapitel so zu schreiben, dass es unabhängig von den übrigen gelesen werden kann. Ein Profi oder „Fortgeschrittener“ muss also nicht von der ersten bis zur letzten Seite vorgehen. Für einen

** Zwar gibt es Ankündigungen anderer Autoren, die mit dem gleichen Thema beschäftigt sind. Es werden wohl mehrere Bücher zur OOo-Basic-Programmierung erscheinen, was ich persönlich nicht schlimm finde, da jedes Buch niemals alle Aspekte für alle Interessierten abdeckt und niemals sämtliche Programmiermöglichkeiten beschreiben kann.

effektiven Einstieg in die zentralen Kapitel 5 bis 8 (das Objektmodell von Calc, Writer, Draw, Impress und Base) empfiehlt es sich allerdings, einen Blick auf das grundlegende Kapitel 1 (der Sprachkern und die Organisation der Makros) zu werfen.

Zu diesem Buch

Den Sprachkern von OOo-Basic zu kennen ist unerlässlich. Er wird in Kapitel 1 ausführlich beschrieben – immerhin nimmt er fast ein Fünftel des Buches ein.

Dialoge mit ihren differenzierten Möglichkeiten, Benutzereingaben abzufangen, werden in Kapitel 2 behandelt. Auch die Dynamik der Dialoge spielt eine große Rolle in diesem Kapitel.

In Kapitel 3 werden die allgemeine Theorie und der Aufbau der Befehle erläutert, in Kapitel 4 beschreibe ich den Zugriff auf Elemente, die in jedem der Programmteile identisch sind, also: Öffnen, Schließen, eine neue Datei anlegen, drucken und überprüfen, in welchem Programm ich mich befinde.

Kapitel 5 bis 8 widmen sich jeweils einem der Programmteile Calc, Writer, Draw und Impress und Base. Ich habe mich für diese Reihenfolge entschieden, da ich vermute, dass die meisten Programme für Calc geschrieben werden. Sicherlich nimmt die Steuerung des Zeichenprogramms Draw und des Präsentationstools Impress nur einen geringen Raum ein. Base habe ich deshalb ans Ende der Viererreihe platziert, weil es erst in OOo 2.0, beziehungsweise StarOffice 8.0 eingeführt wurde.

Ab Kapitel 9 werden unterschiedliche Beispielanwendungen beschrieben, wie sie in Firmen oder im Privatbereich Einsatz finden könnten.

Der Nachteil, der aus diesem Aufbau erwächst, liegt auf der Hand: Einige Themen überschneiden sich und werden im Buch an mehreren Stellen besprochen. Oder es sind Querverweise auf andere Kapitel zu finden.

Mein Wissen

Es wäre vermessen und verlogen, wenn ich nicht die Quellen benennen würde, in denen ich eine Reihe von Hinweisen und Informationen gefunden habe. Bei all diesen Autoren möchte ich mich recht herzlich bedanken – ohne ihr profundes Wissen wäre dieses Buch niemals zustande gekommen beziehungsweise ich hätte sicherlich noch viele Monate tüfteln und suchen müssen, um die Programmiertechniken selbst erlernen zu können.

- Andrew Pitonyak: OpenOffice.org Macros Explained. Hentzenwerke Publishing – Das erste und bislang beste Buch zu OOo-Basic auf dem Markt. Eine Fülle von Beispielen und profunden Erklärungen zur Programmierung.
- http://docs.sun.com/db/817-1826-10 – Eine pdf-Datei, die zum Download bereitsteht, leider nicht vollständig und mit einigen Fehlern.

- http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html liefert die komplette Dokumentation – leider ohne Beispiele sondern lediglich eine immense Ansammlung an Befehlen.
- Von der Seite http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html kann eine über 1000 Seiten umfassende Dokumentation heruntergeladen werden, wie man mit Java OOo steuert.

Weitere Hinweise finden Sie natürlich unter http://api.openoffice.org/

- Ralf Nebelo: Tipps zur Programmierung von StarOffice 6, die wirklich helfen. Linux-Magazin 05/2002 und 07/2002
- Bächinger Consulting http://www.bcwin.ch
- OpenOffice.org deutschsprachig http://de.openoffice.org
- Inzwischen gibt es bereits die ersten deutschsprachigen Newsgroups und Foren zu OOo (Stand Juni 2005, ohne Garantie auf Vollständigkeit):
- Das deutsche OpenOffice.org-Portal: http://www.ooo-portal.de/
- Eine OpenOffice.org-Distribution http://www.openoffice-cd.de/
- Deutsches Forum rund um OpenOffice.org: http://de.openoffice.info
- OpenOffice-CD-Forum: http://www.openoffice-cd.de
- Das OpenOffice.org- und StarOffice-Forum http://www.industrie24.com
- Vescon OpenOffice Support-Forum: http://openoffice.vescon.com/
- Das StarOffice- & OpenOffice.org-Forum auf WinTotal: http://www.wintotal-forum.de
- Peter Herzogs Spotlight.de StarOffice-Forum: http://www.spotlight.de
- Open-Craft.org: Freie Software für freie Handwerker: http://www.open-craft.org
- Forum der Akademie für Management und Technik http//www.amt-wiesbaden.de (diese Seite wird zurzeit überholt)
- Das StarOffice- & OpenOffice.org-Forum: http://supportforum.sun.com

Die Beispiele dieses Buches

Beim Schreiben hatte ich oft den Wunsch, zu einigen Kapiteln viel mehr zu schreiben. Dies begann bereits bei der Frage nach dem Inhalt und der Form. Gerne hätte ich im Buch sämtliche Objekte mit ihren Eigenschaften und Methoden aufgelistet. Das hätte aber den Umfang des Buches gesprengt. Gerne hätte ich dem Buch eine CD beigefügt mit dem kostenfreien Programm OOo, sowie sämtlichen Beispieldateien, wie auch die Liste der Objekte, Eigenschaften und Methoden. Dies hätte allerdings einen höheren Ladenpreis bedeutet, gegen den wir uns – Autor und Verlag – letztlich entschieden haben. Deshalb können Sie sämtliche Beispieldateien von meiner Homepage http://www.compurem.de herunterladen. Sie finden den Link unter AUTORENTÄTIGKEIT | BÜCHER. In den Ordnern finden Sie die gleichen Dateien, die das Buch begleiten. Da die Beispiele von Kapitel 6

nicht mit der Reihenfolge der Dateinamen übereinstimmten habe ich dort auf den Dateinamen verwiesen, unter dem Sie den entsprechenden Code finden können. Einige der Dateien verlangen bestimmte Datei- oder Ordnernamen. Sie müssen möglicherweise geändert werden. Ich habe versucht, sie zentral in einer Konstanten unterzubringen, so dass sie leicht geändert werden können.

Die Hinweise in diesem Buch

Folgende Tabelle gibt eine Übersicht zu den im Buch verwendeten Schreibkonventionen:

Element	**Beispiel**	**Formatierung**
Dateinamen, mit oder ohne Pfad	*C:\Eigene Dateien\Dinos.xls*	*Kursiv*
Menüs	DATEI | DRUCKEN	KAPITÄLCHEN
Spezielle Bezeichnungen, die im Text mitgelesen werden können, Punkte auf Dialogblättern, Steuerelemente, Beschriftungen, Titel, Namen von Symbolleisten, Services	„True“, „Laufweite“, „Standard“, der Service „com.sun.star.sheet.SpreadsheetDocument“ …	in „Anführungszeichen“
Tasten	<Strg>+<A>	<In spitzen Klammern>
Programmiercode, Schlüsselwörter und Variablennamen	`i = i + 1`	`Nichtproportionalschrift`

Um Ihnen die Orientierung in diesem Buch zu erleichtern, habe ich den Text in bestimmte Funktionsabschnitte gegliedert und diese durch entsprechende Hinweise gekennzeichnet. Folgende Symbole finden Verwendung:

Hinweis

Manches ist von besonderer Bedeutung und verdient darum auch, besonders hervorgehoben zu werden. Solche Hinweise sind sehr nützlich; damit erreichen Sie schneller das Ziel.

Tipp

Manches geht ganz leicht. Wenn man nur weiß wie. Tipps und Tricks finden Sie in den Abschnitten, bei denen dieser Text steht.

Achtung

Achtung, mit diesem Hinweis wird eine Warnung angezeigt. An der markierten Stelle sollten Sie besonders achtsam sein.

Beispiel

Gerade im ersten Kapitel wird das theoretisch Erläuterte an verschiedenen Beispielen erläutert. Sie sind durch dieses Icon gekennzeichnet.

Ein Dankeschön ...

geht an den Lektor des Buches, Fernando Schneider, der mich schnell mit Informationen versorgt hat, die ich benötigte. Ebenso an Monika Kraus vom Hanser-Verlag, die wie immer sehr genau den Satz kontrolliert hat. Und natürlich an die Korrektorin, Frau Sandra Gottmann, die viele Fehler aufgespürt hat, die mir entgangen sind.

Ein weiteres Dankeschön geht auch an die Firma Sun, die mir kostenlose StarOffice-Versionen zur Verfügung gestellt hat und mir eine Reihe an Informationen zukommen ließ.

Ebenso gilt mein Dank den Autoren, die im Internet ihr Wissen über OOo-Basic bereitstellen. Und besonders an Herrn Andrew Pitonyak, dessen Buch mir eine große Hilfe war. Ohne ihr Wissen und ihre Kenntnisse wäre dieses Buch nicht zustande gekommen.

… übrigens

Gerne können Sie den gesamten Code ansehen, testen, verändern und für eigene Zwecke weiter benutzen. Ich habe auch kein Problem damit, wenn Sie dieses Buch als Ideensammlung für eigene Projekte verwenden.

Da ich jedoch als freier Trainer, Autor und Programmierer von Aufträgen lebe, wie ich sie hier beschrieben habe, würde ich mich natürlich auch über das Zustandekommen einer gemeinsamen Zusammenarbeit freuen.

Und nun wünsche ich viel Freude beim Lesen, beim Nachdenken und beim Lösen von Aufgaben mit OOo-Basic.

Dr. René Martin

München, August 2005

Über Kritik, Anregungen und Vorschläge freue ich mich sehr.

Rene.Martin@compurem.de

Besuchen Sie auch meine Homepage:

http://www.compurem.de

In diesem Kapitel wird die Entwicklungsumgebung beschrieben. Es wird gezeigt, wie Sie Makros erstellen, testen und speichern können. Ebenso wird der Sprachkern beschrieben, zu dem Variablen, Verzweigungen, Schleifen und das Abfangen und Suchen von Fehlern gehören.

1 Die Entwicklungsumgebung und der Sprachkern

1.1 Wo befinden sich die Makros?

Makros werden in OOo-Basic an Bibliotheken gebunden. Diese sind Teile von Dokumenten. Es ist nicht möglich, lauffähige Basic-Prozeduren außerhalb von OOo zu starten. Ebenso kann man mit OOo keine exe-Dateien erstellen. Zwar kann man OOo „von außen“, das heißt mit Programmiersprachen wie beispielsweise Java, VS.NET, C++ oder anderen starten, aber mit dem „reinen“ OOo-Basic ist es nicht möglich, Programme zu erstellen, die außerhalb von OOo existieren.

Zum Erstellen, Üben und Testen empfiehlt es sich, ein leeres Dokument abzuspeichern und die erstellten Makros daran zu hängen. Zwar könnte man ebenso an die Laufzeitbibliothek von OOo Makros binden – am Anfang sollten Sie jedoch Abstand davon nehmen, da sonst Ihr OOo „zu voll“ wird.

Wenn Sie ein Makro an eine bestimmte Datei gebunden haben, so werden Sie beim nächsten Öffnen gefragt, ob sie den Code ausführen möchten. Dies sollten Sie bejahen.

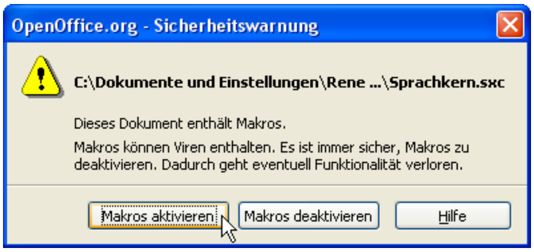


Abbildung 1.1 Die Sicherheitswarnung

1.2 Makros generieren

Sie gelangen in OOo über das Menü EXTRAS | MAKROS | MAKROS VERWALTEN OOO-BASIC (in Version 1.0: EXTRAS | MAKROS | MAKRO) in die Entwicklungsumgebung. Im ersten Schritt müssen Sie angeben, ob Sie ein globales oder lokales Makro erstellen möchten. Das heißt, ob dieses Makro immer, für alle OOo-Sitzungen zur Verfügung stehen soll oder ob es nur an einer Datei hängt. Zum Testen und Üben sollten Sie zu Beginn eine gespeicherte Datei verwenden, dort über

die Schaltfläche „Verwalten“ ein neues Modul und anschließend mit der Schaltfläche „Neu“ einen neuen Namen für das neue Makro festlegen.

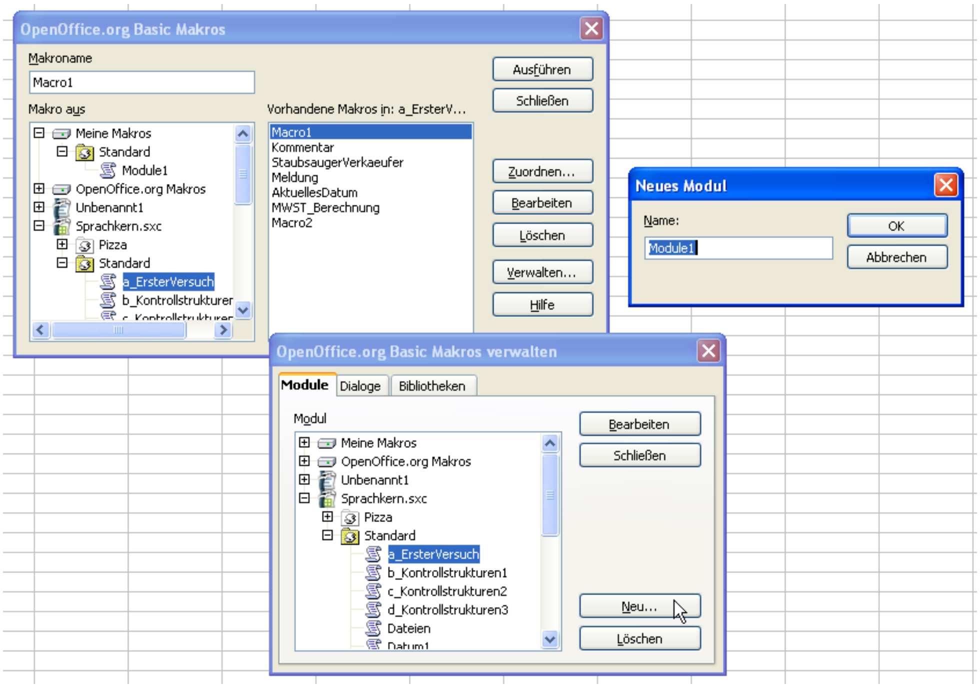


Abbildung 1.2 Die Organisation der Makros

Sie sollten das Makro nicht unbedingt in die Bibliothek „Standard“ speichern. Sie verlieren bei größeren Projekten leicht den Überblick! Legen Sie zuerst eine neue Bibliothek an und erstellen dort Ihre Makros!

Vermeiden Sie doppelte Namen! OOo-Basic führt manchmal zu überraschenden Ergebnissen, wenn eine Variable den gleichen Namen trägt wie eine Prozedur, ein Modul oder eine Bibliothek! So wird beispielsweise nicht ein Ergebnis an die Variable „Test“ übergeben, sondern das Makro „Test“ aufgerufen:

```
Test = MsgBox("Möchten Sie das Dokument wirklich schließen?", _

2 + 48 + 512)
```

Wenn Sie keinen Namen vergeben, dann erhält das Makro den Namen „Macro1“, „Macro2“ … Sinnvoller ist es jedoch, gleich einen vernünftigen Namen zu vergeben.

Die Namensvergabe unterliegt einigen Regeln:

Die Länge des Namens ist unbegrenzt, darf aber nur Buchstaben und Ziffern enthalten, also keine Leerzeichen, Satz- und Sonderzeichen: ,;:-!„§$%&/(){<>}\^. Auch der Bindestrich „-“ darf nicht verwendet werden. Dagegen sind Unterstrich („_“) und Groß- und Kleinschreibung erlaubt. Das erste Zeichen darf keine Ziffer darstellen. Folgender Name ist gültig:

```
Sub Einfach_Nur_Mal_So_1
```

Wird der Name einer Prozedur per Hand eingegeben, müssen Sie das Makro beenden mit dem Befehl:

```
End Sub
```

Viele Programmierer folgen bei der Vergabe von Prozedurnamen der Konvention „Substantiv-Verb“ oder „VerbSubstantiv“. Selbstredend sollte der Name einer Prozedur einen aussagekräftigen Rückschluss auf ihre Funktion erlauben. Und: Verwenden Sie „CamelCasing“, also die Verwendung von Groß- und Kleinschreibung. Das erleichtert das Lesen von Prozedurnamen, beispielsweise „NamenEintragen“, „DateinamenAuslesen“ oder „BenutzerInformationenAnzeigen“.

1.2.1 Makros starten

Normalerweise werden Makros an Schaltflächen, Menüs oder Symbole gebunden. Zum Testen können Sie die Makros aber auch über den Menüpunkt EXTRAS | MAKROS | MAKROS VERWALTEN | OOO. BASIC sehen und dann „starten“ (in OOo 1.1 und StarOffice 7.0 über EXTRAS | MAKROS | MAKRO).

Befinden Sie sich in der Entwicklungsumgebung, dann kann der erste Makro eines Moduls mit der Taste <F5> oder das Symbol „Basic starten“ gestartet werden, alle übrigen müssen zum Testen über EXTRAS | MAKROS | MAKROS ORGANISIEREN | OOO. BASIC (oder EXTRAS | MAKROS | MAKRO) aufgerufen werden.

Möchten Sie zu Testzwecken das Makro vorzeitig beenden, dann fügen Sie den Befehl

```
Exit Sub
```

ein. Die Haltepunkte, Einzelschritte und der Debug-Modus werden in Kapitel 1.12 beschrieben.

1.2.2 Die Entwicklungsumgebung

Die Entwicklungsumgebung IDE („Integrated Development Environment“) ist eine eigenständige Anwendung innerhalb des OOo-Pakets. Sie besteht aus einer Menüleiste, zwei Symbolleisten (der Funktionsleiste und der Makroleiste), dem Fenster, in dem die Makros editiert, das heißt erstellt und verändert werden, dem Beobachtungsfenster („Beobachter“) und dem Fenster, das die Aufrufe überwacht. Dazu mehr weiter unten.

Sie gelangen in bereits existierende Makros aus OOo über den Befehl EXTRAS | MAKROS | MAKROS ORGANISIEREN | OOO-BASIC (in Version 1.0: EXTRAS | MAKROS | MAKRO), von wo aus Sie ein bereits erstelltes Makro „bearbeiten“ können.

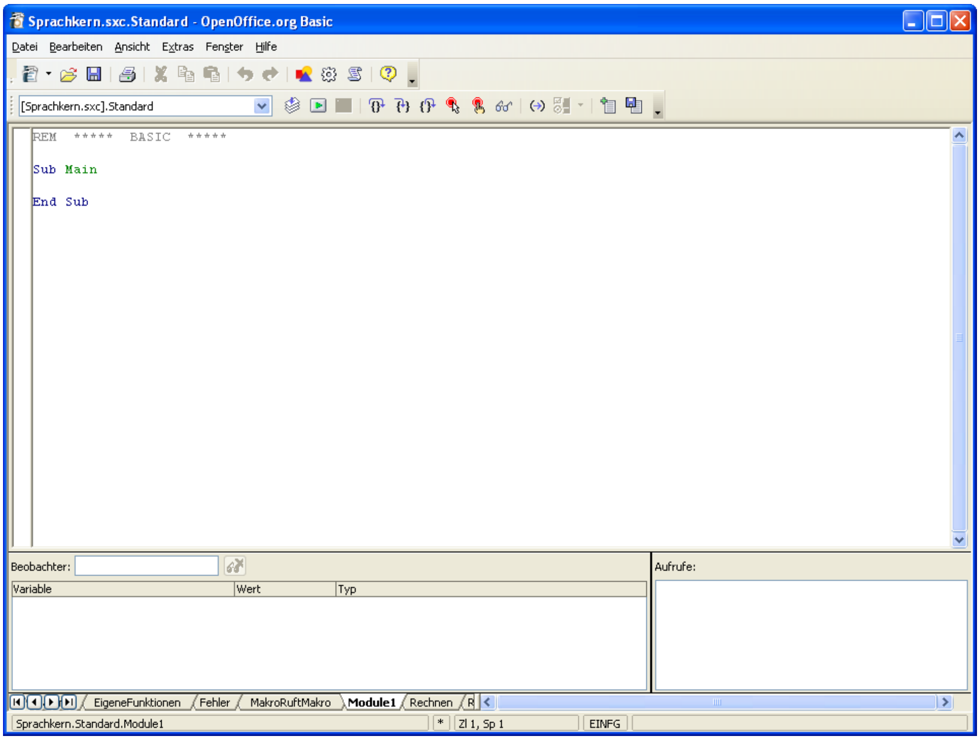


Abbildung 1.3 Die Entwicklungsumgebung

1.2.3 Texteingabe, Textkorrektur, Text verschieben

Die Texteingabe im Codefenster, beziehungsweise die Korrektur des fehlerhaften Codes erfolgen nach den gleichen Regeln wie in der Textverarbeitung: Zu korrigierende Zeichen werden mit der Korrekturtaste <Backspace> oder der Taste <Entf> gelöscht, Text kann an beliebigen Stellen eingefügt werden. Mit der <Einfg>-Taste kann in den Überschreibmodus gewechselt werden: Der Cursor erscheint dann als Rechteck und löscht beim Schreiben die Zeichen. Um Text zu markieren, können Sie die Maus verwenden und mit gedrückter linker Maustaste über den Text ziehen oder mit gedrückter <Shift>-Taste + Pfeiltasten den Cursor über den Text bewegen. Der gesamte

Code, das heißt das ganze Modul, kann auch mit <Strg>+ <A> markiert werden. Zum Kopieren und Ausschneiden kann das Menü BEARBEITEN | KOPIEREN und BEARBEITEN | EINFÜGEN verwendet werden, respektive: BEARBEITEN | AUSSCHNEIDEN und BEARBEITEN | EINFÜGEN. Analog funktionieren die Tastenkombinationen <Strg> + <C>, <Strg> + <X> und <Strg> + <V> oder die Symbole in der Symbolleiste. Man kann auch markierte Textteile mit der Maus verschieben – mit der Drag & Drop-Funktion. Ebenso können Textteile mit der <Strg>-Taste durch Ziehen kopiert werden.

Neue Zeilen und Bewegungen innerhalb der alten

Wenn sich der Cursor innerhalb einer Prozedurzeile befindet und Sie benötigen eine neue Zeile oberhalb der aktuellen, dann müssen Sie den Cursor an den Anfang der Zeile bewegen (<Pos1>) und dort <Enter> drücken. Die vier Pfeiltasten bewegen den Cursor Zeile um Zeile oder Zeichen um Zeichen nach oben, unten, links oder rechts, die Tasten <Bild oben> beziehungsweise <Bild unten> blättern eine Bildschirmseite nach oben oder unten. Die Tabulatortaste erzeugt einen Einzug. Sie kann zwar mit den Tasten <Entf> und der Korrekturtaste gelöscht werden, jedoch nicht mit <Shift>+<Tab>!

Lange Textzeilen

Normalerweise entspricht eine Textzeile einem Befehl. Nach dem Drücken der <Enter>-Taste wird die Eingabe beendet, und ein neuer Befehl kann eingegeben werden. Problemlos können die Anweisungszeilen durch Leerzeilen voneinander getrennt werden – leere Zeilen werden übergangen und haben keinerlei Funktion.

Es kann nun passieren, dass ich eine sehr lange Befehlszeile eingeben muss, zum Beispiel:

```
Anzeigetext = MsgBox("Möchten Sie, dass Ihr Computer zerstört wird?", 3 + 32 +
256, "Odyssee 2000")
```

Ein automatischer Umbruch, wie von der Textverarbeitung bekannt, findet nicht statt: Der Text fließt weiter und weiter nach rechts. Um einen manuellen Umbruch zu organisieren, kann der Text in mehrere Zeilen geteilt werden. Diese werden durch eine Leerstelle (sie ist unbedingt nötig!), der ein Unterstrich am Ende der Zeile folgt, getrennt.

```
Anzeigetext = MsgBox("Möchten Sie, dass Ihr Computer zerstört wird?", _

3 + 32 + 256, _

"Odyssee 2000")
```

Umbruch

Der Unterstrich darf nicht innerhalb von Textteilen stehen. Dies führt zu einer Fehlermeldung wie in folgendem Beispiel:

```
MsgBox "Achtung: _
  Ihr Computer wird in wenigen Sekunden zerstört. _
  Bitte halten Sie sich fest!"
```

Wie dennoch solch ein langer Text, der sich auf mehrere Zeilen erstreckt, erzeugt wird, wird in Kapitel 1.4, „Variablen und Konstanten“, erläutert.

Kurzer Text

Umgekehrt können mehrere logische Programmierzeilen in eine Textzeile geschrieben werden. Dann wird mit einem Doppelpunkt („:") getrennt. Wie beispielsweise folgende drei Befehlszeilen:

```
Msgbox "selamat datang": Beep: MsgBox "Das war Balinesisch."
```

Davon ist in der Regel abzuraten, weil der Code sehr schnell unübersichtlich wird.

Groß- und Kleinschreibung

An (fast) keiner Stelle unterscheidet OOo bei der Code-Eingabe zwischen Groß- und Kleinschreibung. Folgende drei Makros sind äquivalent:

```
SUB ALLES_WAS_ICH_BRAUCHE
   MSGBOX "Ich brauche mindestens " & FORMAT(5000000, "#,##0.00") & _
      " € für mein Glück!"
END SUB

sub alles_was_ich_brauche
   msgbox "Ich brauche mindestens " & format(5000000, "#,##0.00") & _
      " € für mein Glück!"
end sub

Sub Alles_Was_Ich_Brauche
   MsgBox "Ich brauche mindestens " & Format(5000000, "#,##0.00") & _
      " € für mein Glück!"
End sub
```

Abbildung 1.4 Mein bescheidener Wunsch

Diese drei Makros laufen nicht in einem Modul – trotz unterschiedlicher Groß- und Kleinschreibung werden sie als ein Makro identifiziert!

An zwei Stellen wird jedoch zwischen Groß- und Kleinschreibung unterschieden:

- Steuerelemente auf Dialogen müssen im Code exakt so geschrieben werden, wie sie benannt wurden.
- (Interne) Konstanten verlangen exakte Schreibweise, also beispielsweise:

```
lngSearchFlags = com.sun.star.frame.FrameSearchFlag.CREATE + _
   com.sun.star.frame.FrameSearchFlag.ALL
```

und nicht:

```
lngSearchFlags = com.sun.star.frame.FrameSearchFlag.Create + _
   com.sun.star.frame.FrameSearchFlag.All
```

1.2.4 Kommentare und optische Gliederungen

Zu einem gut strukturierten Programm gehören Kommentare. Sie erleichtern anderen Programmierern, damit schnell weiterzuarbeiten und sich schnell einen Überblick zu verschaffen. Sie können für sich selbst Notizen in den Code schreiben, die an keiner Stelle erscheinen. Ebenso sollten Sie Informationen über größere Projekte in den Code schreiben: Erstellt von, Auftraggeber, Versionsnummer, Erstelldatum …

Kommentare können überall eingefügt werden: Sie werden mit einem Apostroph (') eingeleitet, der innerhalb einer Zeile stehen kann:

```
MsgBox "Das ist ein weites Feld."  'Dies ist von Fontane!
```

oder am Anfang einer Zeile:

```
' Nun folgt ein Meldungsfenster:
MsgBox "Ja, Effi, das hab ich. Das ist wirklich ein weites Feld."
' Dies war das Meldungsfenster.
```

Kommentare können ebenso am Beginn einer Zeile durch ein „rem" (remark) eingeleitet werden:

```
Rem Nun folgt ein Meldungsfenster:
MsgBox "Ja, Effi, das hab ich. Das ist wirklich ein weites Feld."
Rem Dies war das Meldungsfenster.
```

Dieses `rem` wirkt etwas altmodisch und erfordert mehr Tipparbeit als Apostrophe. Sonst gibt es keinen Grund, auf `rem` zu verzichten.

Während des Programmierens kann mit „rem“ oder Apostroph eine Zeile ausgeschaltet werden – besonders wenn sie später noch verwendet wird.

Kommentieren Sie blockweise! Kommentieren Sie alles, was Sie programmieren! Auch wenn es auf den ersten Blick nach viel Arbeit aussieht, hilft es Ihnen, doch im Nachhinein schnell einen Überblick über alte oder fremde (von anderen Programmierern geschriebene Codeteile zu erhalten.

Schreiben Sie in ein Modul zu Beginn Ihren Namen, das Erstellungsdatum und das Datum der letzten Änderung. Dann sind Sie sicher, dass Sie die Informationen parat haben, wenn Sie herausfinden müssen, ob Sie in der aktuellen Version programmieren.

Kommentarfarben

Kommentare erscheinen in grauer Schrift, was Sie im Menü EXTRAS | OPTIONEN | OOO | DARSTELLUNG im Bereich „Basic Syntaxhervorhebung“ ändern können. Wenn Sie es unbedingt verändern müssen, dann sollten Sie nicht Schwarz nehmen, denn die Kommentare sollen auffallen. Kommentare werden nicht abgearbeitet, sondern dienen zur Erläuterung, so dass der Programmierer bestimmte Codestellen entweder schnell findet oder Erläuterungen zum Code hat.

Auch Rot ist eine schlechte Farbe für Kommentarzeilen, denn in Rot erscheinen Zahlen und Zeichenketten. Blau ist für Operatoren und reservierte Ausdrücke vorbehalten, grün für Bezeichner. Und schließlich gibt es noch Farben, die das Auge nicht so gerne mag ... Vielleicht bleiben wir doch bei Grau.

Ein beliebtes Gestaltungsmittel, das Ordnung schafft, sind Kommentare in Kästen mit „*“, „-“ oder „=“. Etwa so:

```
'**************************************************

'*                                                *

'*      Und nun für unsere Fontane-Freunde        *

'*                                                *

'**************************************************


   MsgBox "Ja, Effi, das hab ich. Das ist wirklich ein weites Feld."


'**************************************************

'*                                                *

'*             Heute: Effi Briest                 *

'*                                                *

'**************************************************
```

Zugegeben: Solch ein gestalterisches Werk erfordert (Tipp-)Arbeit, strukturiert allerdings sehr gut das Programm.

Einrücken von Zeilen

Eine weitere Gliederungsmöglichkeit ist das Einrücken von Zeilen. Dadurch werden Ebenen gekennzeichnet, was bei Verschachtelungen hilfreich ist und die Übersichtlichkeit erhöht. Sie sollten generell einrücken, um bei längeren Programmen den Überblick nicht zu verlieren.

In diesem Buch wurde an einigen, wenigen Stellen darauf verzichtet, da der Satzspiegel nicht genügend Platz ließ.

Das folgende Beispiel zeigt, wie Sie strukturiert programmieren sollten.

```
Sub StaubsaugerVerkaeufer
   ' ===============================================
   ' = Der Verkäufer gibt ein, wie viele Staubsauger =
   ' = Tag verkauft hat.                              =
   ' = Waren es weniger als fünf, so wird er          =
   ' = gescholten, ab fünf bis zehn Staubsauger       =
   ' = erhält er eine Prämie von 100,00 EURO.         =
   ' = Zwischen zehn und zwanzig eine Prämie von      =
   ' = 200,00 EURO, über zwanzig sogar 500,00 EURO.   =
   ' = Dies wird angezeigt.                           =
   ' ===============================================
   Dim intStaubsauger As Integer
   Dim strText As String
   ' ---- Die Variablen werden deklariert       ----

   intStaubsauger = InputBox("Wie viele Staubsauger?")
   ' ---- Die Abfrage nach der Anzahl der Staubsauger
```

```
   If intStaubsauger < 5 Then
   ' ---- 1. Fall: Anzahl der Staubsauger
      strText = "Fauler Hund!"

   Else
   ' ---- Falls mehr als 5 verkauft wurden
   strText = "Sie erhalten "
      If intStaubsauger < 10 Then
      ' ---- 2. Fall: zwischen 5 und 10
         strText = strText & _
         "100,00 EURO Provision!"

      Else
      ' ---- 3. Fall: zwischen 10 und 20
         If intStaubsauger < 20 Then
            strText = strText & _
            "200,00 EURO Provision!"

         Else
         ' ---- 4. Fall: mehr als 20
            strText = strText & _
            "500,00 EURO Provision!"
         End If
      End If
   End If
   MsgBox strText, 64
End Sub
```

1.3 Eingabe und Ausgabe

1.3.1 Meldungsfenster (Messagebox)

Sicherlich ist eine der Hauptaufgaben der EDV, Daten von einer bestimmten Stelle zu holen, sie weiterzuverarbeiten, neue Daten einzugeben und abzuspeichern. Deshalb soll als Erstes das Meldungsfenster vorgestellt werden.

Für die Ausgabe steht die Möglichkeit zur Verfügung, in ein Textdokument zu schreiben. Dies wird in Kapitel 1.10.3 (Textdateien) und Kapitel 6 (Writer) beschrieben. Oder man kann sich das Ergebnis anzeigen lassen. Dafür existiert das aus OOo bekannte Meldungsfenster.

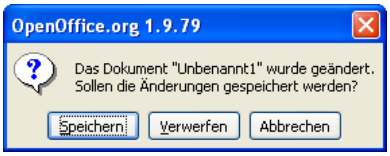


Abbildung 1.5 Ein Meldungsfenster

Der Befehl hierfür lautet:

```
MsgBox
```

Das Meldungsfenster hat folgende Syntax:

```
MsgBox(Text As String, [Type As Integer], [Dialogtitle As String]
As Integer
```

Die in eckigen Klammern geschriebenen Befehle stellen Optionen zur „Verfeinerung" des Meldungsfensters zur Verfügung.

Text

„Text" ist die Meldung und muss angegeben werden. Wird der Text direkt (und nicht als String-Variable) eingegeben, so muss er in Anführungszeichen "" stehen, wobei die ersten Anführungszeichen durch mindestens ein Leerzeichen von der `MsgBox` getrennt sein müssen. Es kann auf folgende Art geschrieben werden:

```
MsgBox "Ich weiß nicht, was soll es bedeuten"
MsgBox ("Ich weiß nicht, was soll es bedeuten")
```

Abbildung 1.6 Das einfache Meldungsfenster

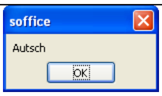
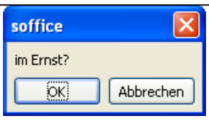
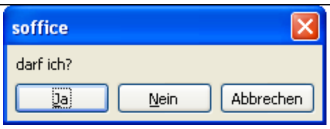
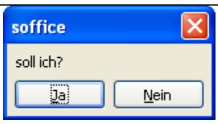
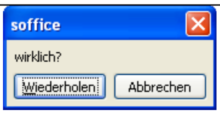
Es existiert ein Unterschied zwischen beiden Schreibweisen; er wird weiter unten erläutert.

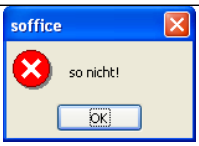
Type

Um verschiedene Schaltflächen erscheinen zu lassen, kann nach dem Komma eine Zahl erscheinen. Beispielsweise:

```
MsgBox "Ich weiß nicht, was soll es bedeuten", 2
```

Tabelle 1.1 Folgende Schaltflächen stehen im Meldungsfenster zur Verfügung:

Typwerte	Beschreibung	Aussehen
0	OK-Schaltfläche	
1	OK- und Abbrechen-Schaltfläche	
2	Abbrechen- und Wiederholen-Schaltfläche	
3	Ja-, Nein-, und Abbrechen-Schaltfläche	
4	Ja- und Nein-Schaltfläche	
5	Wiederholen- und Abbrechen-Schaltfläche	

Typwerte	Beschreibung	Aussehen
16	Stopp-Symbol	
32	Fragezeichen-Symbol	
48	Erklärungssymbol	
64	Informationssymbol	
0	Erste Schaltfläche ist vorbelegt	
256	Zweite Schaltfläche ist vorbelegt	
512	Dritte Schaltfläche ist vorbelegt	

Und nun Beispiele: Das Meldungsfenster

```
MsgBox "Ich weiß nicht, was soll es bedeuten"
```

soll verfeinert werden mit den drei Schaltflächen „Ja“, „Nein“, „Abbrechen“. Dazu kann der Code

```
MsgBox "Ich weiß nicht, was soll es bedeuten", 3
```

verwendet werden.

Abbildung 1.7 Eine andere Meldungsbox

Um nun Kombinationen aus den vorgegebenen Möglichkeiten (mehrere Schaltflächen plus Symbole) zu erzeugen, wird die Summe gebildet. Um beim nachträglichen Überarbeiten nicht den Überblick zu verlieren, kann statt der einfachen Zahl auch die Summe der Zahlen stehen (oder Sie arbeiten mit Konstanten – dies wird in Kapitel 1.4.4 erläutert). Also statt:

```
MsgBox "Ich weiß nicht, was soll es bedeuten", 291
```

kann auch geschrieben werden:

```
MsgBox "Ich weiß nicht, was soll es bedeuten", 3 + 32 + 256
```

Übergabe an eine Variable

Soll nun überprüft werden, welche der Schaltflächen der Benutzer gedrückt hat, so kann das Meldungsfenster an eine Variable übergeben werden. Dann muss allerdings der Text des Meldungsfensters in Klammern stehen:

```
Heine = MsgBox("Ich weiß nicht, [...]", 3)
```

Je nach gedrückter Schaltfläche wird eine andere Zahl zurückgegeben (im obigen Fall an `Heine`).

Tabelle 1.2 Die Rückgabewerte der MsgBox:

Rückgabewerte	**Beschreibung**
1	OK
2	Abbrechen
4	Wiederholen
5	Ignorieren
6	Ja
7	Nein

Mittels dieser Werte kann abgefangen werden, was der Benutzer angeklickt hat, und dann können weitere Schritte eingeleitet werden. Folgende Beispiele führen zum selben Ergebnis:

```
Heine1 = MsgBox("Ein Märchen aus uralten Zeiten", 562)

Heine1 = MsgBox("Ein Märchen aus uralten Zeiten", 2 + 48 + 512)
```

Titel

Der dritte Parameter hinter dem Komma gibt an, was in der Titelleiste stehen soll. Beispiel:

```
MsgBox "Das kommt mir nicht aus dem Sinn.", 64 + 4, "Die Heimkehr"
```

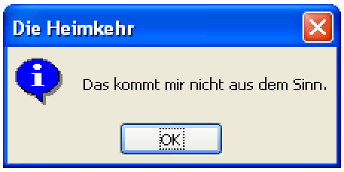


Abbildung 1.8 Meldungsfenster mit Titel und Informationssymbol

Das Meldungsfenster hat sicherlich mehrere Funktionen: Zum einen dient es als „Hinweistafel", um während der Programmierung zu sehen, „ob sich etwas tut", das heißt, um während der Laufzeit Zwischenergebnisse abzufangen und anzuzeigen. Alternativ kann für „MsgBox" der Befehl `Print` verwendet werden:

```
Print "Ein Märchen aus uralten Zeiten"
```

Oder Sie verwenden das Beobachtungsfenster, das weiter unten beschrieben wird.

Ob Sie `MsgBox` oder `Print` einsetzen, ist Ihnen überlassen. Jedoch ist `MsgBox` etwas flexibler. Der Programmierer kann den Benutzer an wichtigen Stellen hinweisen, dass ein besonderes Ereignis eintritt. Meldungsfenster können an prekären Stellen erneut fragen, ob die Aktion des Benutzers gewünscht ist, und ihm die Möglichkeit zum Abbrechen geben (zum Beispiel beim Löschen). Oder Sie können ihn über den Stand der Dinge informieren.

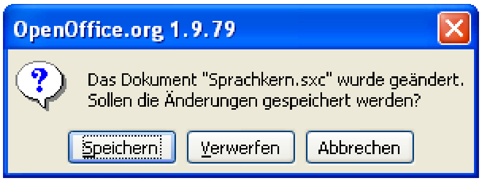


Abbildung 1.9 Ein OOo-Meldungsfenster

Da die Meldungsfenster unflexibel sind (das heißt: der Text kann nicht formatiert werden), empfiehlt es sich, für größere Ausgaben (das heißt für längeren Text, der mehr als 1.024 Zeichen umfasst) oder formatierte Ausgaben einen Dialog zu verwenden. Diese Technik wird im folgenden Kapitel 2 beschrieben.

Zur besseren Lesbarkeit sollten Sie die Zeilen mit der Tabulatortaste konsequent einrücken.

Da die Zahlen für die Buttons schwer lesbar sind, sollten diese Werte – wenn sie häufiger verwendet werden – an Konstanten übergeben werden:

```
Const JA_NEIN_ABBRECHEN As Integer = 3
Const ERKLAERUNGSSYMBOL As Integer = 48
Const ZWEITE_SCHALTFLAECHE = 256

MsgBox "Ein Märchen aus uralten Zeiten", _
   JA_NEIN_ABBRECHEN + ERKLAERUNGSSYMBOL + _
   ZWEITE_SCHALTFLAECHE
```

Beep

Ein häufig verwendeter Befehl im Zusammenhang mit Dialogboxen ist `Beep`. Er macht nichts anderes, als einmal zu piepsen. Um den schläfrigen Benutzer aus seinen Träumen zu wecken ...

1.3.2 InputBox

Das Gegenteil der „MsgBox“ ist die „InputBox“. Über sie kann einfach eine Benutzereingabe entgegengenommen werden. Ihre Syntax lautet:

```
a = InputBox(Msg As String [, Title As String] [, Default As String] [, X_Pos As Integer] [, Y_Pos As Integer])
```

Die Frage

`Msg` ist die Meldung, die in dem Kasten stehen wird:

```
Anzahl = InputBox("Wie viele Bilder enthält der Text?")
```

schreibt den Text „Wie viele Bilder enthält der Text?“ in das Eingabefeld. Der Benutzer trägt etwas in die Eingabezeile ein, klickt auf die OK-Schaltfläche, und der Text wird an die Variable „a“ („Anzahl“) übergeben, mit der nun weitergearbeitet werden kann.

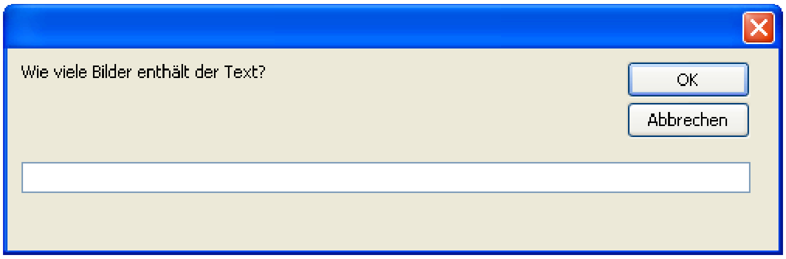


Abbildung 1.10 Die einfache InputBox

Title

Das optionale Feld „Title“ ermöglicht einen Text in der Titelzeile. Also beispielsweise:

```
Anzahl = InputBox("Wie viele Bilder sind im Text?", "Bilder")
```

Die Vorgabe

Soll ein bestimmter Wert im Eingabefeld stehen, so ist dies der Default-Wert (der Vorgabewert), der hinter dem nächsten Komma steht:

```
Anzahl = InputBox("Wie viele Bilder sind im Text?", "Bildermakro", 175)
```

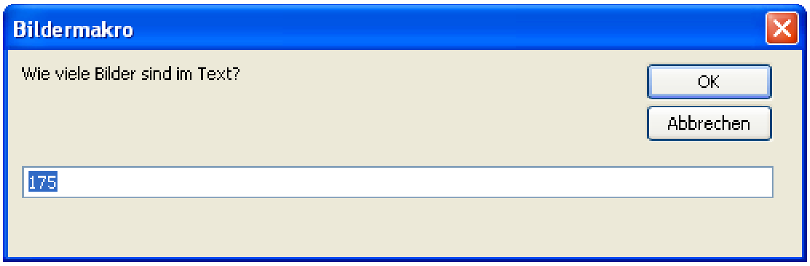


Abbildung 1.11 Die InputBox mit Titel und Vorgabe

Er kann vom Benutzer akzeptiert oder überschrieben werden.

Die Position

Bei der X- und Y-Position kann angegeben werden, ob das Eingabefeld nicht in der Mitte, sondern, gemessen vom linken oberen Bildschirmrand, an einer bestimmten Stelle steht. Stehen keine Werte an dieser Stelle, so erscheint das Eingabefenster in der Mitte des Bildschirms.

Da die InputBox unflexibel ist, sollten Sie nach Möglichkeit darauf verzichten. Eingaben können „eleganter“ über Dialoge entgegengenommen werden – dort kann beispiels-

weise besser zwischen Text und Zahl unterschieden werden, dort können Radiobuttons oder Listenfelder eingebaut werden.

1.4 Variablen und Konstanten

1.4.1 Variablen

Damit vom Benutzer eingegebene Daten wieder ausgegeben werden können, werden sie an Variablen übergeben. Variablen sollten zu Beginn des Programms deklariert werden, müssen aber nicht. Die Variablendeklaration bedeutet, dass festgelegt wird, was die Variable beinhalten kann (und was nicht). Wenn eine Variable nur Zahlen beinhalten darf, ihr dagegen Text übergeben wird, so kommt es zu einer Fehlermeldung.

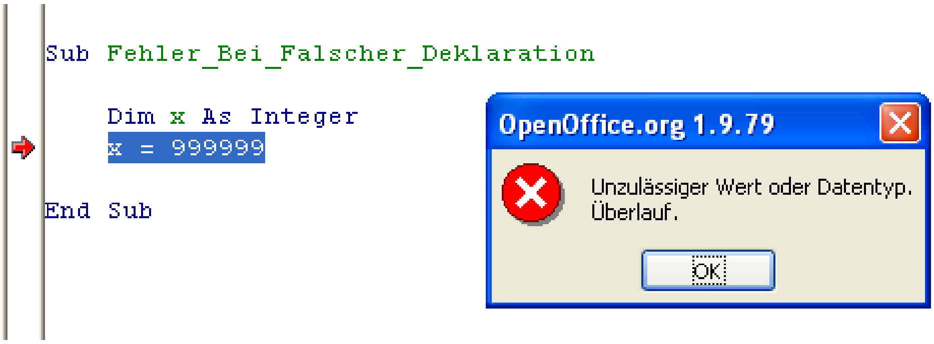


Abbildung 1.12 Die Fehlermeldung bei einer falsch deklarierten Variablen

Die Variable besteht also aus ihrem Namen und ihrem Inhalt. Der Inhalt ist niemals direkt sichtbar, kann aber über eine Reihe von Möglichkeiten sichtbar gemacht werden. Der Name kann sich nicht ändern, der Inhalt dagegen schon.

Das Gleichheitszeichen

Die Funktion `Date` liefert das heutige Datum. Dieses kann an eine Variable übergeben werden. Das heißt, man kann den Inhalt der Variablen sichtbar machen, indem man ihn ausgibt. Dafür dient das Gleichheitszeichen. Rechts vom Gleichheitszeichen steht immer der (alte) Wert, links immer die Variable, die damit gefüllt wird. Einer Variablen mit Namen „Kassiopeia“ kann das heutige Datum übergeben werden, indem Sie die Zuordnung treffen:

```
Kassiopeia = Date
```

Nicht umgekehrt. Über das Meldungsfenster wird angezeigt:

```
Sub AktuellesDatum
   Dim Kassiopeia As Date
   Kassiopeia = Date
   MsgBox Kassiopeia
End Sub
```

Das liefert folgendes Ergebnis:

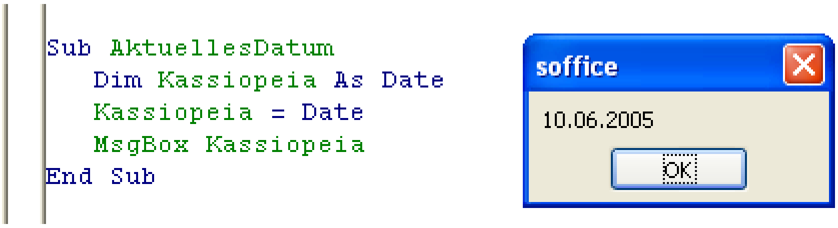


Abbildung 1.13 Das Ergebnis von Kassiopeia

„Kassiopeia" ist also Träger des aktuellen Datums.

Auf der rechten Seite des Gleichheitszeichens kann ein Wert oder ein Text stehen, ebenso gut kann sich dort aber auch eine Berechnung befinden. Statt

```
KoelnischWasser = 4711
```

könnte auch stehen:

```
KoelnischWasser = 4700 + 11
```

oder:

```
KoelnischWasser = 673 * 7
```

Der Inhalt der Variablen

Wenn Sie den Inhalt einer Variablen ändern möchten, so können Sie dies jederzeit tun. Die Variable hat stets den zuletzt übergebenen Wert. Also:

```
KoelnischWasser = 4709
KoelnischWasser = 4710
KoelnischWasser = 4711
```

wird im Meldungsfenster

```
MsgBox KoelnischWasser
```

4711 ausgegeben.

Alte Werte können aber auch geändert werden, indem auf der rechten Seite des Gleichheitszeichens der alte Wert verwendet, geändert und dem Wert auf der linken Seite übergeben wird:

```
KoelnischWasser = 4701
' Noch ist KoelnischWasser = 4701
KoelnischWasser = KoelnischWasser + 10
' Nun wird zum alten Wert KoelnischWasser 10
' addiert, das heißt, die Variable
' KoelnischWasser enthält nun den Wert 4711!
```

Variablendeklaration

Zwar könnte man in Basic auf eine Variablendeklaration verzichten, dennoch sollte man dies – aus mehreren Gründen – nicht tun. Da Basic jede Zeile beim Kompilieren auf Fehler überprüft, findet die Programmiersprache bei der expliziten Deklaration sofort Tippfehler in den Variablennamen, wenn `Option Explicit` am Anfang des Moduls eingegeben wurde. Und Tippfehler sind nicht selten!

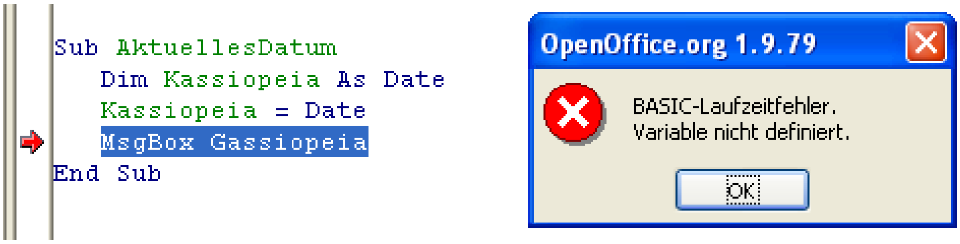


Abbildung 1.14 Eine fehlerhafte Variable

Verwenden Ihre Programme eine große Anzahl an Zahlen, so hat eine Variablendeklaration den Vorteil, den Arbeitsspeicher nicht so sehr zu belasten wie mit Texten. Denn bei Zahlen können Sie den reservierten Speicherbereich festlegen.

Die Namen der Variablen

Die Namen der Variablen müssen mit einem Buchstaben beginnen, dürfen Buchstaben (außer den Umlauten), Ziffern und den Unterstrich enthalten und dürfen nicht mit einem Basic-Schlüsselwort identisch sein.

Explizite Deklaration

Variablen können explizit oder implizit deklariert werden. Eine explizite Deklaration steht immer am Anfang des Programms und folgt der Syntax:

```
Dim [Variablenname] As [Variablentyp]
```

Implizite Deklaration

Wenn keine Variablendeklaration erfolgen soll, muss keine solche Zeile vorhanden sein – die Variable taucht irgendwann innerhalb des Programmiercodes auf und verschwindet einfach wieder. Lautlos.

Verwenden Sie immer die explizite Variablendeklaration! Und zwingen Sie sich, mit `Option Explicit` jede verwendete Variable zu deklarieren!

Der Wertebereich der Variablen

Bevor eine Variable deklariert wird, ist es wichtig, den Inhalt zu kennen, mit dem sie gefüllt wird. Überschreiten die Werte den definierten Bereich, so kommt es zu Laufzeitfehlern. Umgekehrt ist es sinnvoll, die Deklaration so klein wie möglich zu halten, um den Speicherbereich sinnvoll zu nutzen.

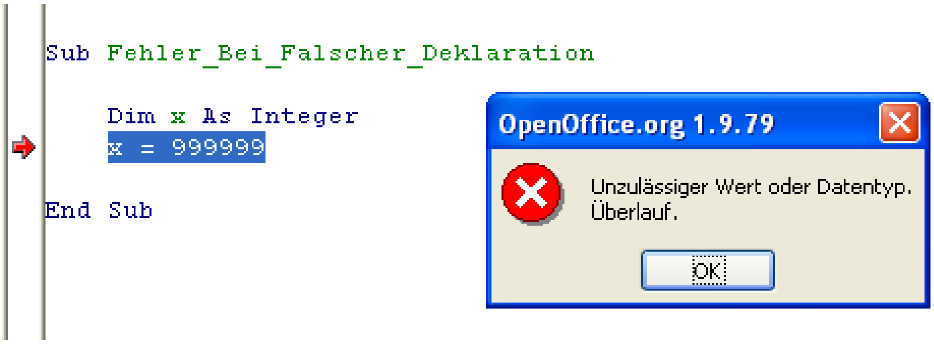


Abbildung 1.15 Eine Fehlermeldung

Datentypen

Folgende Datentypen stehen zur Verfügung: Ganzzahlen, Dezimalzahlen, Währungen, Zeitangaben und Zeichenketten. Im Einzelnen sieht das so aus:

Boolean

Boolean: Dieser Datenwert kann nur die beiden Werte „Wahr“ und „Falsch“, also „True“ und „False“ verwalten. Er benötigt 2 Byte Speicherplatz. Es entspricht `True` = 0 und `False` = 1.

Integer

Integer: Damit werden Ganzzahlen zwischen –32.768 und 32.767 verwaltet. Neben der Deklaration:

```
Dim IntegerZahl As Integer
```

ist auch:

```
Dim IntegerZahl%
```

möglich: Das %-Zeichen am Ende der Variablen zeigt an, dass es sich hierbei um eine Integer-Zahl handelt:

```
Dim Zaehler%
```

Long

Long: Mit Long (dieser Typ wird häufig „Long Integer“ genannt) werden lange Ganzzahlen verwaltet, das heißt Zahlen zwischen –2.147.483.648 und 2.147.483.647.

Beispiel für eine Long-Zahl:

```
Dim Kontonummer As Long
```

oder alternativ:

```
Dim Kontonummer&
```

Hierbei dient das & als Typenkennzeichen.

Single

Single: Mit Single werden Dezimalzahlen verwaltet, die zwischen $-1{,}401 \times 10^{-45}$ und $3{,}402 \times 10^{38}$ liegen. Neben der Deklaration:

```
Dim Gewicht As Single
```

gibt es auch hier eine Variante mit einem Typenkennzeichen:

```
Dim Gewicht!
```

Double

Double: Auch Double verwaltet alle möglichen Fließkommazahlen, nämlich solche, die zwischen $-4{,}94 \times 10^{-234}$ und $1{,}798 \times 10^{308}$ liegen. Dafür wird diesem Variablentyp ein Speicherbereich von 8 Byte eingeräumt:

```
Dim Lichtjahre As Double
```

Oder mit dem Typenkennzeichen #:

```
Dim Lichtjahre#
```

Date

Date: Date verwaltet, wie der Name schon sagt, Datumsangaben. Nicht alle, sondern „nur“ zwischen dem 1.1.100 und dem 31.12.9999. Historiker, die mit früheren Daten (Antike?) arbeiten, und Zukunftsforscher, die noch weiter planen, müssen auf andere Programme ausweichen oder sich mit „Krücken“ helfen. Apropos Historiker: Natürlich gab es vor Christi Geburt weder den julianischen noch den gregorianischen Kalender. In OOo-Basic liegen aber keine anderen Kalender vor – weder der julianische noch der jüdische oder islamische. Vorgegebene Umrechnungsmodi fehlen.

```
Dim Datumsangabe As Date
```

Currency

Currency: Damit werden Währungsangaben mit 15 Stellen vor dem Komma und bis zu vier Nachkommastellen verwaltet, in einem Bereich von $–9{,}22 \times 10^{-14}$ und $9{,}22 \times 10^{14}$. Neben einer Deklaration wie beispielsweise

```
Dim Einnahmen As Currency
```

kann auch mit dem AT-Zeichen @ deklariert werden:

```
Dim Einnahmen@
```

String

String: Damit werden Texte, das heißt beliebige Zeichenketten bis maximal 65.535 Zeichen, verwaltet und übergeben. Das Dollar-Zeichen $ ist hier das typische Typenkennzeichen:

```
Dim Wohnort As String
```

```
Dim Wohnort$
```

Mehrere Variablen

Werden mehrere Variablen deklariert, dann nicht:

```
Dim Heinrich1, Heinrich2, Heinrich3 As Integer
```

und auch nicht:

```
Dim Heinrich1 As Integer, Heinrich2, Heinrich3
```

sondern:

```
Dim Heinrich1 As Integer, Heinrich2 As Integer, Heinrich3 As Integer
```

Tabelle 1.3 Die Variablen und ihre Deklarationsmöglichkeiten:

Datentyp	**Variablentyp**	**Typen-kennzeichen**	**Bereich**	**Speicherplatz (in Bytes)**	**Präfix**
Ja/Nein	Boolean		0 (False) und -1 (True)	2	f oder bln
Ganzzahlen	Integer	%	-32.768 bis 32.767	2	int
	Long	&	-2.147.483.648 bis 2.147.483.647	4	lng
Dezimalzahlen	Single	!	$-3{,}402823 \times 10^{38}$ bis $3{,}402823 \times 10^{38}$	4	sng
	Double	#	$-1{,}797693 \times 10^{324}$ bis $1{,}797693 \times 10^{324}$	8	dbl
	Currency	@	$-9{,}22 \times 10^{14}$ bis $9{,}22 \times 10^{14}$	8	cur
Datumszahlen	Date		1.1.100 bis 31.12.9999	8	dtm
Text	String	$	64.000 Zeichen	10 + Länge der Zeichenkette	str
Wechselnde Typen	Variant		jeder numerische Wert im Bereich Double, jeder String	22 + Länge der Zeichenkette	var

Variablen sollten ein Präfix besitzen, an dem ihr Typ erkennbar ist. Diese Konvention, die von der Firma Gregory Reddick & Associates, einer Unternehmensberatungsfirma von Microsoft, herausgegeben wurde, ist nicht verbindlich. Allerdings arbeiten sehr viele Programmierer damit. Diese Konvention stellt eine Möglichkeit der Standardisierung für VBA-Programmierung dar und wird manchmal in anderen Basic-Programmen verwendet. Sie müssen sich natürlich nicht daran halten, sollten sich aber, wenn Sie zu mehreren an einem Projekt programmieren, über die Benennung der Variablen einig sein. Beispiele:

```
Dim fSchalter As Boolean
Dim intZaehler As Integer
Dim lngKontonr As Long
Dim sngGewicht As Single
Dim dblLichtjahre As Double
```

```
Dim curEinkommen As Currency
Dim dtmEndTermin As Date
Dim strEingabeText As String
```

In anderen Büchern, Zeitschriftenartikeln oder Internetseiten finden Sie häufig ähnliche Konventionen, beispielsweise:

```
Dim iZaehler As Integer
Dim sEingabeText As String
```

1.4.2 Umwandlungsfunktionen

In einigen (seltenen) Fällen ist es wichtig, explizit von einem Datentyp in einen anderen umzuwandeln. Dafür stehen folgende Funktionen zur Verfügung:

Tabelle 1.4 Umwandlungsfunktionen

Funktion	Rückgabewert
CBool	Boolean
CDate	Date
CDbl	Double
CDateFromISO	wandelt eine Zeichenkette, die ein Datum im ISO-Format darstellen in ein Datum um: CDateFromISO("20050407") liefert 07.04.2005
CDateToISO	wandelt ein Datum in eine Zeichenkette im ISO-Format um: CdateToISO("07.04.2005") liefert 20050407
CInt	Integer
CLng	Long
CSng	Single
CStr	String

1.4.3 Datenfelder (Arrays)

Häufig kommt es vor, dass der Programmierer eine große Anzahl von Variablen benötigt, die miteinander in Beziehung stehen, beispielsweise die zwölf Monate. Mit einem Typenkennzeichen (§, $, %, & ...) oder einem Schlüsselwort (Integer, Long, Single, String ...) kann eine Matrix definiert werden. Damit die Variablen nicht einzeln benannt werden müssen (`strMonat1`, `strMonat2`, `strMonat3` ... `strMonat12`), kann eine Dimension in Klammern eingefügt werden. Zum Beispiel:

```
Dim strWochentage(6) As String

Dim strJuenger(11) As String

Dim intKundennummer(123) As Integer
```

OOo-Basic beginnt mit der Zählung bei 0, das heißt 7 Wochentage, 12 Jünger oder (maximal) 124 Kunden mit einer Nummer werden auf 6, 11 beziehungsweise 123 reduziert. Soll dagegen explizit bei 1 angefangen werden, so ist der Befehl

```
Option Base 1
```

am Anfang des Moduls, das heißt vor der ersten Prozedur, nötig. Damit wäre

```
Dim strWochentage(7) As String

Dim strJünger(12) As String

Dim intKundennummer(124) As Integer
```

möglich. Man könnte auch explizit festlegen, wo die Datenfelder beginnen:

```
Dim strWochentage(1 To 7) As String
```

Danach könnten die Variablen übergeben werden mit:

```
strWochentage(1) = "Montag"

strWochentage(2) = "Dienstag"

strWochentage(3) = "Mittwoch"

[...]
```

Deklarieren eines dynamischen Datenfeldes

Bei der Deklaration eines dynamischen Datenfeldes können Sie die Größe des Datenfeldes verändern, während der Code ausgeführt wird. Verwenden Sie zur Deklaration eines Datenfeldes eine der Static-, Dim-, Private- oder Public-Anweisungen, und lassen Sie die Klammern leer, beispielsweise:

```
Dim sngDatenfeld() As Single
```

Sie können die ReDim-Anweisung dazu verwenden, ein Datenfeld implizit innerhalb einer Prozedur zu deklarieren. Achten Sie darauf, bei Verwendung der ReDim-Anweisung den Namen des

Datenfeldes richtig zu schreiben. Auch wenn sich die Option Explicit-Anweisung im Modul befindet, wird ein zweites Datenfeld erstellt.

Verwenden Sie in einer Prozedur innerhalb des Gültigkeitsbereichs des Datenfeldes die ReDim-Anweisung zur Änderung der Anzahl von Dimensionen, zur Definition der Anzahl der Elemente und zur Definition der oberen und unteren Grenzen jeder Dimension. Sie können die ReDim-Anweisung beliebig oft verwenden, um das dynamische Datenfeld zu ändern. Dies hat jedoch zur Folge, dass die bestehenden Werte des Datenfeldes verloren gehen. Verwenden Sie `ReDim Preserve`, um ein Datenfeld zu erweitern, ohne dass die vorhandenen Werte im Datenfeld gelöscht werden. Die Obergrenze kann mit der Funktion `UBound`, die Untergrenze (meistens 0) mit `LBound` herausgefunden werden.

Die folgende Anweisung vergrößert z.B. das Datenfeld `sngDatenfeld` um zehn Elemente, ohne die aktuellen Werte der ursprünglichen Elemente zu löschen.

```
ReDim Preserve sngDatenfeld (UBound(sngDatenfeld) + 10)
```

Wenn Sie das Schlüsselwort `Preserve` mit einem dynamischen Datenfeld verwenden, können Sie nur die obere Grenze der letzten Dimension, aber nicht die Anzahl der Dimensionen ändern.

Beispiel: Das folgende Beispiel liest alle Dateien aus einem Ordner ein und speichert sie in einem Datenfeld, das anschließend ausgelesen wird:

```
Sub DatenFeld()
   Const SUCHPFAD As String = "C:\"
   Const DATEI As Integer = 0
   Dim strDateiNamen() As String
   Dim lngDateienAnzahl As Long
   Dim strGefundenerDateiName As String

   lngDateienAnzahl = 0
   strGefundenerDateiName = Dir(SUCHPFAD, DATEI)
   Do While strGefundenerDateiName <> ""

        ReDim Preserve strDateiNamen(lngDateienAnzahl) As String
        strDateiNamen(lngDateienAnzahl) = strGefundenerDateiName
        lngDateienAnzahl = lngDateienAnzahl + 1
        strGefundenerDateiName = Dir
   Loop
   ' -- durchlaufe den Pfad und suche alle Dateien,
```

```
   ' -- die in einem Datenfeld gespeichert werden

   strGefundenerDateiName = ""
   For lngDateienAnzahl = 0 To UBound(strDateiNamen())
      strGefundenerDateiName = strGefundenerDateiName & _
         Chr(13) & strDateiNamen(lngDateienAnzahl)
   Next     ' -- das Datenfeld wird ausgelesen
   MsgBox strGefundenerDateiName, , "Inhalt von " & SUCHPFAD
End Sub
```

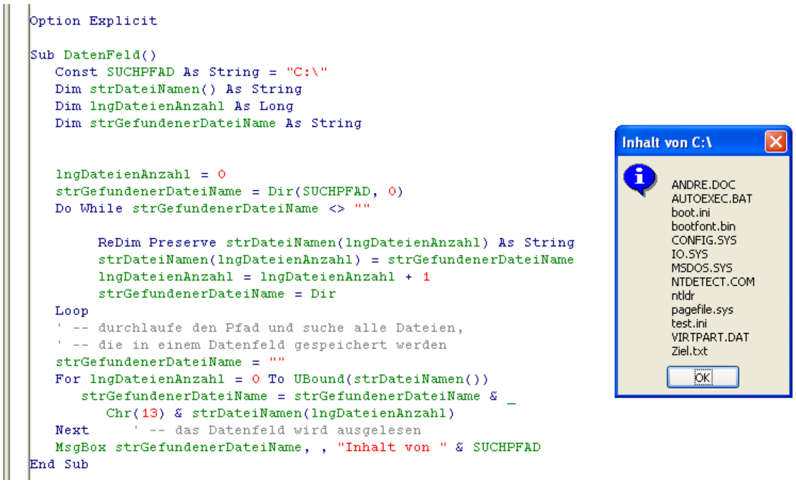


Abbildung 1.16 Ein Datenfeld wird gefüllt und ausgelesen.

1.4.4 Konstanten

Konstanten sind mit Variablen vergleichbar. Sie dienen nicht als Container für unterschiedliche Werte, sondern sind nur Platzhalter, um sich in der Regel Schreibarbeit zu ersparen oder ein Programm besser lesbar zu gestalten. Konstanten werden einmal festgelegt, können dann aber nie mehr geändert werden. Auch sie können lokal oder global deklariert werden. Konstanten werden normalerweise in Großbuchstaben geschrieben.

Die Deklaration funktioniert wie bei den Variablen, nur mit dem Unterschied, dass statt des `Dim` ein `Const` steht. Und der Wert wird festgelegt:

```
[Public] [Private] Const Konstantenname = Wert
```

```
      Sub MWST_Berechnung
         Const MWST = 0.16
         Dim curNetto As Double
         Dim curBrutto As Double
   curNetto = InputBox("Wie lautet der Preis?")
   curBrutto = curNetto + curNetto * MWST
   MsgBox "Der Bruttopreis beträgt " & curBrutto
End Sub
```

1.4.5 Lebensdauer von Variablen

Variablen leben prinzipiell nur so lange, wie das Makro auch lebt. Variablen werden mit den Prozeduren geboren, bleiben eine Lebensspanne bei ihnen und sterben mit ihnen. Dennoch kann man ihr Leben verlängern. Angenommen, wir haben eine Bibliothek in einem Dokument. Nennen wir ein Modul darin modPIZZA In diesem Modul befinden sich zwei Prozeduren: modBODEN und modBELAG. modBODEN enthält die Variable strFluessigkeit, die mit „Öl“ gefüllt wird. Ebenso enthält Belag die Variable strFluessigkeit, die dort allerdings mit „passierte Tomaten“ belegt wird. Sollen die beiden Makros die gleiche Variable verwenden, dann muss sie modulweit deklariert werden. Dies geschieht über der ersten Prozedur. Nun kann die erste Prozedur die Variable füllen, die zweite aufrufen, und diese kennt dann den Inhalt der Variablen:

```
Dim strMehl As String

Sub Boden
   Dim strFluessigkeit As String
   strFluessigkeit = "Öl"
   strMehl = "Vollkornmehl"
   MsgBox strFluessigkeit
   Call Belag
End Sub

Sub Belag
   Dim strFluessigkeit As String
   strFluessigkeit = "passierte Tomaten"
```

```
   MsgBox strFluessigkeit
   MsgBox strMehl
End Sub
```

Wird in einem weiteren Modul (nennen wir es „modKUCHEN") das gleiche Spiel noch einmal durchgeführt, dann beeinflussen sich die Variablen nicht:

```
Dim strMehl As String


Sub Boden
   Dim strFluessigkeit As String
   strFluessigkeit = "Milch"
   strMehl = "Weißmehl"
   MsgBox strFluessigkeit
   call Belag
End Sub
Sub Belag
   Dim strFluessigkeit As String
   strFluessigkeit = "Kirschsaft"
   MsgBox strFluessigkeit
   MsgBox strMehl
End Sub
```

Wenn Sie jedoch modulübergreifende Variablen deklarieren möchten, die in mehreren Modulen und Prozeduren verwendet werden, dann müssen diese `Public` deklariert werden:

```
Public curEier as Currency
```

Sie werden in einer Prozedur gefüllt:

```
   curEier = 0.6
```

Und in einer anderen aufgerufen. Damit eine zweite Prozedur aufgerufen werden kann, die den gleichen Namen hat wie die Prozedur innerhalb des Moduls, muss der Modulname davor geschrieben werden. Zwischen Modulname und Prozedurname sitzt als Trennzeichen ein Punkt:

```
   call modKUCHEN.Belag
```

In verschiedenen Modulen deklarierte Variablen beeinflussen sich nicht.

1.5 Rechnen und verknüpfen

1.5.1 Die Grundrechenarten

Wenn an Variablen Zahlen übergeben werden, so können die vier Grundrechenarten +, -, * und / auf die Variablen angewandt werden. Bitte benutzen Sie keinen Doppelpunkt (:) bei der Division, kein „x“ beim Produkt. Die beiden einzigen Zeichen, die für Multiplikation und Division erlaubt sind, sind Stern (*) und Schrägstrich (/). Schließlich bleibt noch eine fünfte Rechenart zu erwähnen: ^ steht für „hoch“, also die Potenz. Im Beispiel

```
x = 2^3
```

erhält x den Wert 8, im Beispiel

```
x = 25 ^ 0.5
```

den Wert 5.

Es gelten die Rechenregeln der Mathematik: Klammer vor Potenz vor Punkt- vor Strichrechnung, also:

```
y = 1 + 2 * 3
```

ergibt 7 für y, dagegen bei

```
y = (1 + 2) * 3
```

wird y der Wert 9 übergeben. Folglich hat z den Wert 9 in

```
z = 27 ^1 / 3
```

aber 3 in der Berechnung:

```
z = 27 ^(1 / 3)
```

Daneben stellt Basic die üblichen (vielleicht aus Calc bekannten) mathematischen Funktionen zur Verfügung, wie `sin` und `cos`, `log` und `exp` ..., aber auch eine Reihe finanzmathematischer Funktionen, wie `ddb`, `fv`, `pmt`, `ppmt` und andere, die hier nicht behandelt werden.

2.50 statt 2,50 im Code!

Vergessen Sie nicht: Dezimalstellen werden im Quellcode mit einem Punkt („.“) geschrieben, nicht mit Komma (,). Mit Prozentwerten kann nicht gerechnet werden – sie werden übergangen:

```
curBuchMWSt = curBuchPreis * 7%
```

liefert den siebenfachen Wert. Richtig wäre:

```
curBuchMWSt = curBuchPreis * 0.07
```

Tabelle 1.5 Die Funktionen im Einzelnen

Funktionsname	Beschreibung	Beispiel
exp	die natürliche Exponentialfunktion	exp(1)
log	der natürliche Logarithmus	log(1)
sqr	Wurzel	sqr(2)
sin	Sinus	sin(PI/2)
cos	Kosinus	cos(0)
tan	Tangens	tan(0)
abs	Absolutbetrag	abs(-1.2) liefert 1,2
int	liefert die Vorkommazahl durch Abrunden	int(-1.2) liefert -2
fix	liefert die Vorkommazahl durch Abschneiden der Nachkommaziffern	fix((-1.2) liefert -1
sgn	liefert -1 bei negativen Zahlen, 0 bei 0 und 1 bei positiven Zahlen	sgn(-1.2) liefert -1
rnd	ermittelt einen Zufallswert zwischen 0 und 1	
randomize	initialisiert den Zufallsgenerator	

Weitere mathematische Funktionen oder Funktionen aus Bereichen Kombinatorik, Statistik … existieren leider nicht – sie müssen nachgebildet werden (siehe Kapitel 1.8). Hierzu zählen beispielsweise: Arcsin, Arccos, Arctan, Sinhyp, Coshyp, Tanhyp, Arcsinhyp, Arccoshyp, Arctanhyp, Ln, Log10, Fakultät und viele andere

Ein bekanntes Beispiel ist die Berechnung des Ostersonntags! Die Lösung dieses Problems stammt vom Mathematiker, Astronom und Physiker Carl Friedrich Gauß und sieht wie folgt aus: Die Jahreszahl sei J und J - 1900 sei a. Der Rest von a/19 wird schlicht b genannt. Jetzt wird vom Ausdruck (7*b+1)/19 der ganzzahlige Quotient genommen, der c genannt wird. Mit d wird der Rest von (11*b+4-c)/29 bezeichnet und der Quotient von a/4 mit e. Dann bleibt noch der Rest von (a+e+31-d)/7. Und dieser soll f genannt werden. Daraus folgt, dass für das Osterdatum April die Formel 25 - d - f gilt.

Soll beispielsweise von 2004 der Ostersonntag berechnet werden, so ergeben sich folgende Werte:

```
J = 2004

a = 104

b = REST(104;19) = 9

c = QUOTIENT(7*9+1;19) = 3

d = REST(11*9+4-3;29) = 13

e = QUOTIENT(104;4) = 26
```

```
f = REST(104+26+31-13;7) = 1

Ostern =DATUM(2004;4;25-11-3) = 11.04.2004
```

Analog für das Jahr 2006:

```
a = 106; b = 11; c = 4; d = 5; e = 26; f = 4; Ostern = 16.04.2006
```

```
Sub Ostersonntag()
   Dim intJahreszahl As Integer
   Dim a As Integer, b As Integer, c As Integer
   Dim d As Integer, e As Integer, f As Integer
   Dim g As Integer
   Dim datOsterSonntag As Date

   intJahreszahl = InputBox("Bitte eine Jahreszahl eingeben!")

   a = intJahreszahl - 1900
   b = a Mod 19
   c = (7 * b + 1) \ 19
   d = (11 * b + 4 - c) Mod 29
   e = a \ 4
   f = (a + e + 31 - d) Mod 7
   g = 25 - d - f

   If g > 0 Then
         datOsterSonntag = DateSerial(intJahreszahl, 4, g)
   Else
       datOsterSonntag = DateSerial(intJahreszahl, 3, 31 + g)
   End If
   MsgBox "Ostersonntag fällt im Jahre " & intJahreszahl _
      & " auf den " & datOsterSonntag
End Sub
```

Da OOo-Basic nicht mit negativen Tagen in der Funktion `DateSerial` rechnen kann (das Datum `DateSerial(2005, 4, -4)` kann nicht ermittelt werden), muss am Ende mit einer Verzweigung gearbeitet werden. Sie wird in Kapitel 1.6.2 beschrieben. Die Funktion `DateSerial` wird in 1.6.3 erläutert.

1.5.2 Textverkettung

Ebenso wie Zahlen durch Rechenoperationen miteinander verknüpft werden, können auch Textblöcke verknüpft werden (Concatenation). Zwei (oder mehrere) Zeichenketten können mit dem Zeichen + oder dem Zeichen & verbunden werden. Dies ist bei der Ausgabe längerer Texte nötig, da innerhalb der Textzeile nicht mit dem Unterstrich (_) umbrochen werden darf. Beispielsweise:

```
Sub Heine1
   Dim strZeile1 As String, strZeile2 As String
   strZeile1 = "Das Fräulein stand am Meere"
   strZeile2 = "Und seufzte lang und bang,"
   MsgBox strZeile1 & strZeile2
End Sub
```

Abbildung 1.17 Heine1

Es erscheint geschickter, zwischen den beiden Additionsmöglichkeiten zu differenzieren, das heißt, für Zahlenadditionen + zu verwenden, für Stringverkettungen dagegen &. Die Ausgabe im Meldungsfenster ist noch unschön, da der Text ohne Zeilenwechsel „addiert“ wird. Eine Zeilenschaltung kann mit dem ANSI-Code des Absatzzeichens (des Et-Zeichens oder Alineazeichens ¶) eingefügt werden. Es hat den Code 13 und wird mit Chr$(13) eingefügt. Ebenso könnte der Zeilenwechsel verwendet werden. Er hat den ANSI-Code 10. Also:

```
Sub Heine2
   Dim strZeile1 As String, strZeile2 As String
   strZeile1 = "Das Fräulein stand am Meere"
   strZeile2 = "Und seufzte lang und bang,"
   MsgBox strZeile1 & chr$(10) & strZeile2
End Sub
```

Abbildung 1.18 Heine2

Zu lange Zeichenkette

Die letzte Zeile der Stringverkettung wird sehr lang. Um sie abzukürzen, stehen mehrere Möglichkeiten zur Verfügung. Ein Zeilenumbruch innerhalb der Programmierzeile wird mit einem _ eingeleitet und aus Übersichtsgründen in der zweiten Zeile eingerückt (wie im obigen Beispiel). Man kann die Textteile auch nacheinander addieren, also:

```
Sub Heine3
   Dim strZeile As String
   strZeile = "Es rührte sie so sehre"
   strZeile = strZeile & chr(13)
   strZeile = strZeile & "Der Sonnenuntergang"

   MsgBox strZeile
End Sub
```

Neben dem ¶-Zeichen werden manchmal das Anführungszeichen ("), der Apostroph (') und der Tabulator benötigt. Hier die Liste:

Tabelle 1.6 Erzeugbare Strings

Zeichen	Code
<Tabulator>	Chr$(9)
<Enter>	Chr$(13)
Zeilenwechsel	Chr$(10)

Der Text "Mein Fräulein!" wird in einer Meldungsbox ausgegeben mit:

```
Sub Heine4
   Dim strZeile As String
   strZeile = """Mein Fräulein! sein Sie munter,"
```

```
    strZeile = strZeile & Chr(13)
    strZeile = strZeile & "Das ist ein altes Stück;"
    strZeile = strZeile & Chr(13)
    strZeile = strZeile & "Hier vorne geht sie unter"
    strZeile = strZeile & Chr(13)
    strZeile = strZeile & _
       "Und kehrt von hinten zurück." & Chr(34)

    MsgBox strZeile
End Sub
```

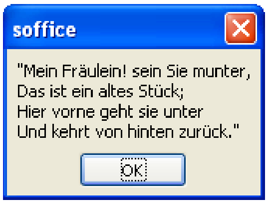


Abbildung 1.19 Heine4

Es funktioniert auch, wenn statt eines Anführungszeichens zwei Anführungszeichen im Text stehen. Sie werden von Basic als ein darzustellendes Anführungszeichen interpretiert:

Direkt in den Text eingegebene Anführungszeichen (") führen zu einer Fehlermeldung, da Basic davon ausgeht, dass die Zeichenkette nun zu Ende ist. Achten Sie beim Korrekturlesen von Text auch auf fehlende Leerzeichen. Ausgegebener Text muss immer innerhalb der Anführungszeichen (" ") stehen.

1.5.3 Textfunktionen

Die Zeichenketten können über String-Funktionen weiter bearbeitet werden. Angenommen, es sollen mögliche Leerstellen gelöscht werden, die der Benutzer vor oder nach dem eingegebenen Text eingefügt hat, so können hierfür die Funktionen `Ltrim` oder `Rtrim` verwendet werden, die Leerzeichen vor oder nach dem Text löschen. Die Funktion `Trim` löscht sowohl davor als auch danach. Mit `Left` und `Right` werden Zeichenketten beschnitten, mit `Instr` kann eine Zeichenkette gesucht werden, `Len` ermittelt die Länge.

Wer braucht das? Stellen Sie sich eine Zeichenkette vor, die Vor- und Nachname enthält. Sie soll auf den Vornamen oder auf den Nachnamen reduziert werden. Über eine InputBox gibt der Benutzer seinen ganzen Namen ein, der dann in Vor- und Zuname zerlegt wird.

```
Sub NamenZerlegen1
   Dim strGname As String, strVorname As String, _
      strNachname As String
   Const strT1 = "Du heißt mit Vornamen "
   Const strT2 = "und Dein Nachname lautet: "
   strGname = InputBox("Wie heißt Du?")
   strVorname = Left(strGname, InStr(strGname, " "))
   'Filtert den Vornamen, indem das
   'Leerzeichen gesucht wird
   strNachname = Right(strGname, Len(strGname) - _
      InStr(strGname, " "))
   ' Da Right von rechts abschneidet, muss die
   ' Differenz zwischen Gesamtlänge und Position von
   ' links gebildet werden.
   MsgBox strT1 & strVorname & strT2 & strNachname
End Sub
```

Abbildung 1.20 Der Name wird in Vor- und Nachname zerlegt.

Ebenso, wie nicht definierte Variablen kein Kennzeichen benötigen, da sie vom Typ Variant sind (der Variablen mit Namen ZAHL kann der Text „Goethe“ übergeben werden), benötigt Basic nicht das $-Zeichen.

Statt der etwas umständlichen Methode „Schneide von rechts so viele Zeichen weg, wie der ganze Name abzüglich der Länge des Vornamens ist“, könnte auch etwas anders geschrieben werden:

```
strNachname = Right(strGname, Len(strGname) - Len(strVorname))
```

Oder noch eleganter mit der Funktion `Mid`. Sie löst ab einer bestimmten Position einen Text heraus. Das Beispiel funktioniert auch so:

```
Sub NamenZerlegen3
   Dim strGname As String, strVorname As String, _
      strNachname As String
   Const strT1 = "Du heißt mit Vornamen "
   Const strT2 = "und Dein Nachname lautet: "
   strGname = InputBox("Wie heißt Du?")
   strVorname = Mid(strGname, 1, InStr(strGname, " "))
   strNachname = Mid(strGname, _
      InStr(strGname, " ") + 1)
   MsgBox strT1 & strVorname & strT2 & strNachname
End Sub
```

Es geht noch leichter: Die Funktion `Split` zerlegt eine Zeichenfolge in Komponenten, wobei die Leerstelle als Standard-Trennzeichen verwendet wird. Das Ergebnis wird an einen Array übergeben.

```
Sub NamenZerlegen4
   Dim strGName As String
   Dim strTeilnamen() As String
   Dim strT1 As String
   Dim strT2 As String
   strGName = InputBox("Wie heißt Du?")
   strTeilnamen = Split(strGName)

   strT1 = "Du heißt mit Vornamen " & strTeilnamen(0)
   strT2 = " und Dein Nachname lautet: " & _
      strTeilnamen(1)

   MsgBox strT1 & strT2
End Sub
```

Umgekehrt setzt die Funktion `Join` einen String wieder zusammen, wobei sie ebenfalls einen Array verlangt. Im folgenden Beispiel werden die einzelnen Komponenten wieder zusammengefügt:

```
Sub NamenZerlegenUndZusammensetzen
   Dim strTeilnamen() As String
   Dim i As Integer
   Dim strGName As String

   strGName = InputBox("Wie heißt Du?", "Alle Namen")
   strTeilnamen = Split(strGName)
      For i = 0 To UBound(strTeilnamen)
       strGName = Join(strTeilnamen, "-")
   Next

   MsgBox strGName
End Sub
```

Folgende (wichtige) String-Funktionen stehen Ihnen in Basic zur Verfügung. Die übrigen finden Sie in der Hilfe:

Tabelle 1.7 Die String-Funktionen

Funktionsname	Bedeutung	Beispiel
Left	schneidet eine bestimmte Anzahl von Zeichen von links ab	Left("August von Platen", 4) ergibt "Augu"
Right	schneidet eine bestimmte Anzahl von Zeichen von rechts ab	Right("August von Platen", 3) ergibt "ten"
Mid	schneidet eine bestimmte Anzahl von Zeichen aus der „Mitte“ heraus, das heißt ab einer bestimmten Position	Mid("August von Platen", 4) ergibt "ust von Platen", Mid("August von Platen", 8, 3) ergibt "von"
InStr	überprüft, ob eine Zeichenfolge innerhalb einer Zeichenkette vorhanden ist, und gibt die Position an	InStr("August von Platen", "v") ergibt 8, InStr("August von Platen", "u") ergibt 2, InStr("August von Platen", "y") ergibt 0
Ltrim	löscht Leerzeichen am Anfang eines Strings	LTrim(" August von Platen ") ergibt ("August von Platen ")
Rtrim	löscht Leerzeichen am Ende eines Strings	RTrim(" August von Platen ") ergibt (" August von Platen")
Trim	löscht Leerzeichen am Anfang und En-	Trim(" August von Platen ") ergibt

Funktionsname	Bedeutung	Beispiel
	de eines Strings	("August von Platen")
Len	ermittelt die Länge einer Zeichenkette	Len("August von Platen") ergibt 17
Chr	wandelt einen ASCII-Code in einen String um	Chr(13) ergibt ¶, Chr(64) ergibt @
Asc	wandelt einen String in die entsprechende Zahl des ASCII-Codes um	Asc("A") ergibt 65
Str	wandelt eine Zahl in einen String um	Str(99) = "99"
Val	wandelt einen String in eine Zahl um	Val("104") = 104
Lcase	wandelt eine Zeichenkette in Kleinbuchstaben um	Lcase("August von Platen") ergibt "august von platen"
Ucase	wandelt eine Zeichenkette in Großbuchstaben um	Ucase("August von Platen") ergibt "AUGUST VON PLATEN"
StrComp	„vergleicht“ zwei Zeichenketten, das heißt, es wird überprüft, welche zuerst im Alphabet steht. Ist die erste Zeichenkette „größer“ als die zweite, wird –1 zurückgegeben, im umgekehrten Fall 1. Sind beide gleich: 0. Ist einer der beiden Strings leer, so wird Null übergeben.	StrComp("August", "August") ergibt 0, StrComp("August", "Platen") ergibt -1, StrComp("von", "Platen") ergibt 1
Space	gibt eine Folge von Leerzeichen aus	"August" & Space(2) & "Platen" ergibt "August Platen"
Split	trennt eine Zeichenfolge und liefert einen Array	Split("August von Platen") ergibt "August" "von" "Platen"
Join	setzt eine Zeichenfolge zusammen	

Praktische Beispiele ergeben sich bei den Textfunktionen, wenn Dateinamen in einer bestimmten Art angezeigt oder generiert werden. Der Dateiname der aktuellen Datei wird ausgelesen und modifiziert:

```
Sub DateiName
   Dim strDatei As String
   Dim strTeile() As String
   strDatei = ThisComponent.URL
   MsgBox strDatei

   strDatei = Mid(strDatei, 9)
```

```
   MsgBox strDatei
   strTeile = Split(strDatei, "%20")
   strDatei = Join(strTeile, " ")
   MsgBox strDatei
   strTeile = Split(strDatei, "/")
   strDatei = Join(strTeile, "\")
   MsgBox strDatei
End Sub
```

Der Dateiname hat beispielsweise folgende Gestalt:

```
file:///C:/Dokumente%20und%20Einstellungen/
   Rene%20Martin/Eigene%20Dateien/StarBasic/ Kapitel01/Sprachkern.sxc
```

Zuerst wird das `file:///` entfernt. Anschließend die Zeichen „%20“ durch Leerzeichen ersetzt und im dritten Schritt die „/“ durch „\“. Das Ergebnis sieht dann wie folgt aus:

```
C:\Dokumente und Einstellungen\Rene Martin\Eigene
   Dateien\StarBasic\Kapitel01\Sprachkern.sxc
```

Es geht mit den beiden Funktionen `ConvertFromUrl` und `ConvertToUrl` eleganter:

```
   strDatei = ThisComponent.URL
   MsgBox strDatei
   strDatei = ConvertFromUrl(strDatei)
   MsgBox strDatei
   strDatei = ConvertToUrl(strDatei)
   MsgBox strDatei
```

1.6 Kontrollstrukturen

1.6.1 Vergleiche und Konjunktoren

Neben den mathematischen Funktionen und den Textfunktionen existieren Vergleichsoperatoren und Verknüpfungskonjunktoren:

Zwei Texte oder Zahlen werden mit folgenden Operatoren verglichen:

Tabelle 1.8 Mathematische Verknüpfungen

Operator	Beschreibung
=	Gleich
<	Kleiner als
>	Größer als
<=	Kleiner gleich
>=	Größer gleich
<>	Ungleich

Ebenso können zwei logische Operatoren miteinander verglichen werden:

Tabelle 1.9 Logische Operatoren

Operator	Beschreibung
Not	Negation
And	Logisches Und
Or	Logisches Oder (das eine oder das andere)
Xor	Exklusives Oder (entweder das eine oder das andere)
Eqv	Logische Äquivalenz (beide Ausdrücke sind True oder beide sind False)
Imp	Implikation (der erste Ausdruck ist True, der zweite False – nur dann wird False zurückgegeben)

Die Verknüpfungsmöglichkeiten können in einer Tabelle aufgelistet werden:

Tabelle 1.10 Logische Verknüpfungen und ihre Ergebnisse

Operator	True, True	True, False	False, True	False, False
And	True	False	False	False
Or	True	True	True	False
Xor	False	True	True	False
Eqv	True	False	False	True
Imp	True	False	True	True

1.6.2 Verzweigungen

If ... Then ... Else ... End If

Vielleicht ist die häufigste Verzweigung die If-Verzweigung.

```
If Bedingung Then Anweisung1 [Else Anweisung2]
```

Nennen wir sie Supermarkt-Verzweigung: Wenn das Marzipan billiger als 2,00 € ist, kaufe ich 10 Kilo.

```
If Marzipanpreis < 2,00 € Then einkaufe 10 Kilo
```

Und was, wenn nicht? Wenn das Marzipan billiger als 2,00 € ist, kaufe ich 10 Kilo, sonst nur ein Päckchen:

```
If MPreis < 7,00 € Then einkaufe 10 Kilo Else einkaufe 1 Päckchen
```

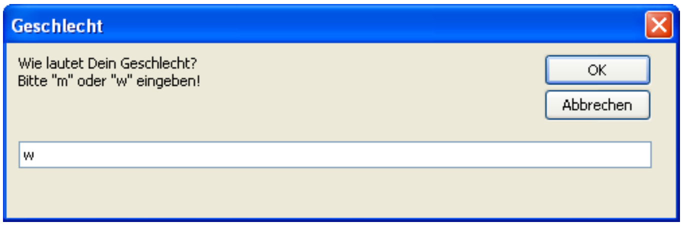
Dies muss nicht in einer Zeile stehen, sondern kann auch untereinander geschrieben werden. Dann beachten Sie aber, dass die If-Verzweigung mit einem `End If` geschlossen werden muss. Und vergessen Sie das `Then` nicht! Die Syntax:

```
If Bedingung Then
   Anweisung1
   [Anweisung2]
   [Anweisung3]

[Else Anweisung2]
   [Anweisung1]
   [Anweisung2]
   [Anweisung3]
End If
```

Durch ein Makro wird ein Benutzer oder eine Benutzerin nach seinem oder ihrem Geschlecht gefragt. Gibt er oder sie ein „m“ oder ein „w“ ein, dann wird der Nachname erfragt, und er oder sie wird begrüßt. Gibt die Benutzerin oder der Benutzer dagegen etwas Falsches ein, wird er oder sie beschimpft. Tippt er ein großes „M“ oder ein großes „W“ ein, so werden diese Buchstaben in Kleinbuchstaben verwandelt. Das Makro reagiert auf alle vier Möglichkeiten positiv.

```
Sub Geschlecht_und_Name
```

```
  Dim strGeschlecht As String, strName As String
  Dim strFrage As String

  strFrage = "Wie lautet Dein Geschlecht?" & Chr(13) & _
     "Bitte ""m"" oder ""w"" eingeben!"
  strGeschlecht = InputBox(strfrage, "Geschlecht")
  strGeschlecht = LCase(strGeschlecht)
  If strGeschlecht = "m" Or strGeschlecht = "w" Then
     strName = InputBox("Wie lautet Dein Name?")
     If strGeschlecht = "m" Then
        MsgBox "Sei mir gegrüßt, lieber " & _
        strName, 48, "Gruß"
     Else
        MsgBox "Sei mir gegrüßt, liebe " & _
        strName, 48, "Gruß"
     End If
  Else
        MsgBox "Ein ""m"" oder ""w""," & _
          " du Walnusshirn!", 16, "Kein Gruß"
  End If
End Sub
```



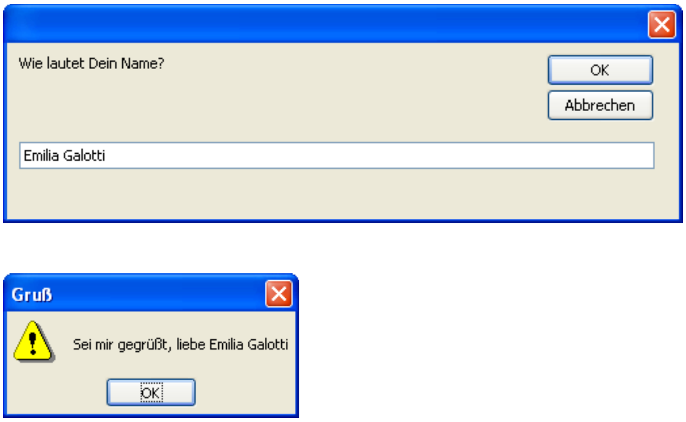


Abbildung 1.21 Frage und Antwort

If-Verzweigungen können ineinander verschachtelt werden.

IIf

Eine (sehr selten verwendete) Verzweigung hat die folgende Syntax:

```
IIf(expr, truepart, falsepart)
```

Sie funktioniert wie die If-Verzweigung – mit dem Unterschied, dass sie etwas kompakter geschrieben wird und als Funktion einen Wert zurückgibt, während bei If verschiedene Anweisungen abgearbeitet werden. Auf `IIf` wird an dieser Stelle nicht weiter eingegangen.

Die Case-Verzweigung

Manchmal sind es zu viele Fälle, die nicht mit hintereinander geschalteten If-Verzweigungen und auch nicht verschachtelten If-Verzweigungen gestaltet werden können oder sollen. Dafür steht ein weiterer Verzweigungstyp zur Verfügung: die Case-Verzweigung.

Ihre Syntax lautet:

```
Select Case Variable
   Case Wert1
      Anweisung1
   Case Wert2
      Anweisung2
   Case Wert3
```

```
      Anweisung3
[...]
   [Case Else]
      [Letzte Anweisung]
End Select
```

Dabei ist zu beachten, dass Vergleichsoperatoren nur mit einem IS-Statement verwendet werden können, beispielsweise:

```
Select Case Variable
Case Is > 1
```

Verschiedene Argumente können durch Kommata getrennt hintereinander geschrieben werden:

```
Case 2, 3, 4
```

Bereiche können mit To zusammengefasst werden:

```
Case 2 To 4
```

Gesucht ist die Anzahl der vergangenen Monate eines Geschäftsjahres. Das Geschäftsjahr fängt im April an, also aus 4 wird 1, aus 5 (Mai) wird 2, ... aus 12 (Dezember) wird 9, aus 1 (Januar) wird 10 ... und aus 3 wird 12.

```
Sub Geschaeftsjahr1
   Dim intMonat As Integer
   Dim intAusgabeMonate As Integer
   Dim strText As String

   intMonat = InputBox("Bitte einen Monat eingeben!")
   Select Case intMonat
      Case 1, 2, 3
         intAusgabeMonate = intMonat + 9
         strText = "Es sind " & intAusgabeMonate & _
            " Monate vergangen."
      Case 4
         intAusgabeMonate = intMonat - 3
         strText = "Es ist " & intAusgabeMonate & _
            " Monat vergangen."
```

```
      Case 5 to 12
         intAusgabeMonate = intMonat - 3
         strText = "Es sind " & intAusgabeMonate & _
            " Monate vergangen."
      Case Else
         strText = "Falsche Eingabe!"
   End Select

   MsgBox strText

End Sub
```

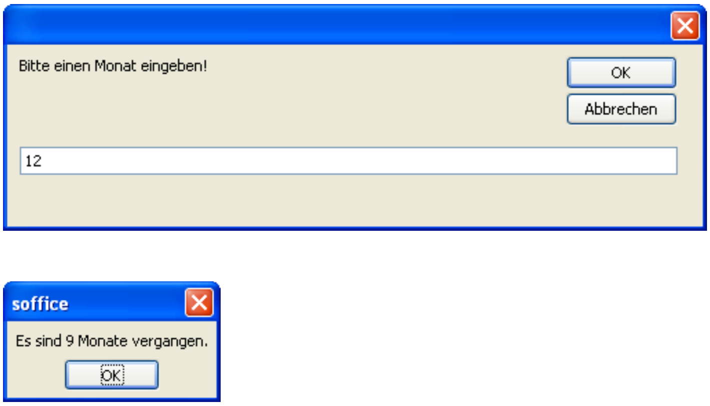


Abbildung 1.22 Die Monate des Geschäftsjahres

Eine andere Variante funktioniert wie folgt:

```
Sub Geschaeftsjahr2
   Dim intMonat As Integer
   Dim intAusgabeMonate As Integer
   Dim strText As String

   intMonat = InputBox("Bitte einen Monat eingeben!")
   Select Case intMonat
   Case 1 To 3
```

```
      intAusgabeMonate = intMonat + 9
      strText = "Es sind " & intAusgabeMonate & _
         " Monate vergangen."
   Case 4
      intAusgabeMonate = intMonat - 3
      strText = "Es ist " & intAusgabeMonate & _
         " Monat vergangen."
   Case Is > 4
      intAusgabeMonate = intMonat - 3
      strText = "Es sind " & intAusgabeMonate & _
         " Monate vergangen."
   Case Else
      strText = "Falsche Eingabe!"
   End Select

   MsgBox strText

End Sub
```

Sie kommen fast immer mit If ... Then ... Else aus. Wird nur eine Bedingung überprüft, dann finden viele Programmierer diese Verzweigung übersichtlicher. Wenn allerdings mehrere Werte gleichzeitig überprüft werden, dann ist Select Case eleganter. Beispielsweise ist:

```
If strMonat = "Januar" Or strMonat = "Februar" Or _
   strMonat = "März" Or strMonat = "April" Then
```

sehr lang und umständlich. Kürzer ist doch:

```
Select Case strMonat
   Case "Januar", "Februar", "März", "April"
```

Aber oft kann die eine oder die andere Verzweigung alternativ verwendet werden.

Wird jedoch nur ein einziger Fall geprüft, dann gestaltet sich eine If-Verzweigung kürzer:

```
Select Case blnDateiVorhanden
   Case False
...
If blnDateiVorhanden = False Then ...
```

Häufig können If-Verzweigungen oder Select Case-Statements synonym verwendet werden.

Noch ein Beispiel (diesmal zur If-Verzweigung):

Der Benutzer soll in ein Eingabefeld sein Geburtsdatum eingeben. Daraufhin wird überprüft, ob er heute Geburtstag hat, was ihm über ein Meldungsfenster mitgeteilt wird:

```
Sub Geburtstag
   Dim datGebDatum As Date
   Dim intTag As Integer
   Dim intMonat As Integer
   Dim lngAnzTage As Long

   datGebDatum = InputBox("Dein Geburtsdatum?")

   intTag = Day(datGebDatum)
   intMonat = Month(datGebDatum)

   If intTag = Day(Date) And _
      intMonat = Month(Date) Then
         MsgBox "Happy Birthday"
   Else
      lngAnzTage = DateValue(Date) - DateSerial(Year(Date), _
         intMonat, intTag)

      If lngAnzTage < 0 Then
         MsgBox "Du hast erst in " &  - lngAnzTage & _
         " Tagen Geburtstag."
      Else
         MsgBox "Du hattest vor " & lngAnzTage & _
         " Tagen Geburtstag."
      End If
   End If
```

```
End Sub
```

Abbildung 1.23 Ein mögliches Ergebnis

Einige Erläuterungen dazu im folgenden Kapitel.

Choose

Während `IIf` sich nur zwischen einer von zwei Auswahlmöglichkeiten entscheiden kann, so kann die Funktion `Choose` aus einer Reihe von Argumenten auswählen. Die Syntax lautet:

```
Choose(Index, Auswahl-1[, Auswahl-2, ... [, Auswahl-n]])
```

Im folgenden Beispiel wählt die Funktion Auswahl zwischen vier Werten aus, die ihr übergeben wurden:

```
Function Auswahl(i As Integer) As String
   Auswahl = Choose(i, "München", "Hamburg", "Berlin", "Köln")
End Function
```

Auch sie gibt – anders als Select Case – direkt einen Wert zurück. Da andererseits keine komplexen Anweisungen verarbeitet werden können, wird im Folgenden nicht näher auf Choose eingegangen.

1.6.3 Datumsfunktionen

Date, Day, Month und Year

Die Variable `datGebDatum` wird vom Typ `Date` deklariert. Sie nimmt das Datum auf, das der Benutzer eingibt. Aus ihr werden die Informationen `Day` und `Month` herausgezogen und mit dem aktuellen Tag und dem aktuellen Monat verglichen. Nur wenn beide zugleich mit dem heutigen Tag und dem heutigen Monat übereinstimmen, hat der Benutzer Geburtstag. `Year` zu vergleichen ist nicht nötig – ja, ist sogar falsch, da das Geburtsjahr ein anderes ist als das aktuelle Jahr.

Das Systemdatum wird mit der Funktion `Date` ermittelt. Analog hierzu arbeiten die Funktionen `Hour`, `Minute` und `Second` beziehungsweise `Time` für die aktuelle Zeit. Aktuelles Datum und Uhrzeit können mit `Now` abgefangen werden. Mit `DateValue` wird der Zahlenwert des Datums ermittelt, mit `TimeValue` der Zahlenwert der Zeitangabe. Die Funktion `Weekday` liefert den Wochentag

(Sonntag = 1, Montag = 2 ... Samstag = 7). Hat der Benutzer nicht Geburtstag, so wird mit der Funktion `DateDiff` die Differenz zwischen dem heutigen Datum und dem Geburtstag des nächsten Jahres berechnet. Sollte der Benutzer aber noch im gleichen Jahr Geburtstag haben, so wird 365 davon abgezogen. Diese Zahl wird danach in einem Meldungsfenster angezeigt.

Tabelle 1.11 Die Datums- und Zeitfunktionen

Funktionsname	Bedeutung	Beispiel
Date	setzt das aktuelle Systemdatum ein oder stellt das Systemdatum um	
Now	gibt das Systemdatum und die aktuelle Systemzeit zurück	
Time	setzt die aktuelle Systemzeit ein oder stellt die Systemzeit um	
DateSerial	gibt die fortlaufende Datumszahl eines Datums zurück: DateSerial(Year, Month, Day)	DateSerial(2005, 12, 24) liefert den 24.12.2005
DateValue	gibt das Datum als Zahl (Long) zurück	DateValue("01.05.2005") liefert den 01.05.2005
TimeSerial	gibt einen fortlaufenden Zeitwert für eine aus Stunden, Minuten und Sekunden bestehende Uhrzeit zurück	TimeSerial(12, 0, 0) liefert 12:00:00
TimeValue	wandelt eine Uhrzeit in eine Zahl (Date) um	TimeValue("11:55") liefert 11:55:00
Day	filtert den Tag aus einem Datum	Day(DateSerial(2005, 12, 24)) liefert 24
Month	filtert den Monat aus einem Datum	Month(DateSerial(2005, 12, 24)) liefert 12
Year	filtert das Jahr aus einem Datum	Year(DateSerial(2005, 12, 24)) liefert 2005
Weekday	gibt eine Zahl zwischen 1 und 7 zurück, die dem Wochentag entspricht: Weekday(Date). Dabei entspricht die 1 Sonntag.	Weekday(DateSerial(2005, 12, 24)) liefert 7 (entspricht Samstag)
Hour	filtert die Stunde aus einer Uhrzeit	Hour(TimeSerial(12, 0, 0)) liefert 12
Minute	filtert die Minutenanzahl aus einer Uhrzeit	Minute(TimeSerial(12, 0, 0)) liefert 0
Second	filtert die Sekundenanzahl aus einer Uhrzeit	Second(TimeSerial(12, 0, 0)) liefert 0

Damit können Sie berechnen, an welchem Tag Sie geboren sind. Und wann Sie in diesem Jahr Geburtstag haben werden.

```
Sub Geburtstagwochentag1
   Dim datGebDatum As Date
   Dim intWochentag1 As Integer
   Dim intWochentag2 As Integer
   Dim strGeburtstagG As String
   Dim strGeburtstagA As String

   datGebDatum = InputBox("Wann bist du geboren?")
   intWochentag1 = Weekday(datGebDatum)
   intWochentag2 = Weekday(DateSerial(Year(Date), _
      Month(datGebDatum), Day(datGebDatum)))
   strGeburtstagG = TagBestimmen(intWochentag1)
   strGeburtstagA = TagBestimmen(intWochentag2)

   MsgBox "Du bist an einem " & strGeburtstagG & " geboren."
   If DateSerial(Year(Date), Month(datGebDatum), _
      Day(datGebDatum)) <= Date Then
   MsgBox "Du hattest dieses Jahr an einem " & _
      strGeburtstagA & " Geburtstag."
   Else
      MsgBox "Du wirst dieses Jahr an einem " & _
      strGeburtstagA & " Geburtstag haben."
   End If
End Sub

Function TagBestimmen(w)
   Select Case w
```

```
        Case 2: TagBestimmen = "Montag"
        Case 3: TagBestimmen = "Dienstag"
        Case 4: TagBestimmen = "Mittwoch"
        Case 5: TagBestimmen = "Donnerstag"
        Case 6: TagBestimmen = "Freitag"
        Case 7: TagBestimmen = "Samstag"
        Case 1: TagBestimmen = "Sonntag"
    End Select
End Function
```

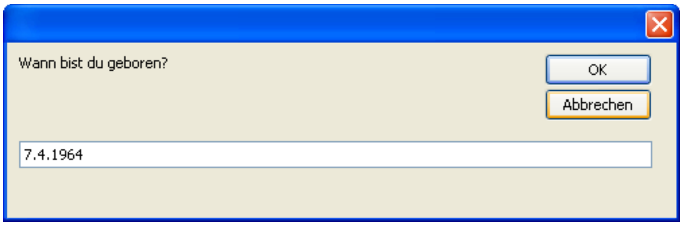


Abbildung 1.24 Die Vorgabe

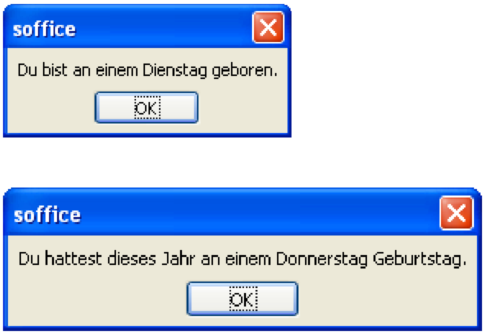


Abbildung 1.25 Die beiden Ergebnisse

Weekday

Einige Erläuterungen: Die Funktion `Weekday(datGebdatum)` berechnet den Wochentag des Geburtstags und gibt eine Zahl zurück (1 steht für Sonntag). Ebenso liefert

`WeekDay(DateSerial(Year(Date), Month(datGebdatum), Day(datGebdatum)))` den Wochentag des Geburtstages des laufenden Jahres (`Year(Date)`). Über eine Case-Verzweigung wird die Zahl in den Wochentagtext umgewandelt. Beim Geburtsdatum des laufenden Jahres wird überprüft, ob dieser vor dem heutigen Tag liegt oder danach. Dazu wird das aus Geburtstag und aktuellem Jahr zusammengesetzte Datum mit dem heutigen Tag verglichen:

```
If DateSerial(Year(Date), Month(datGebdatum), _
   Day(datGebdatum)) <= Date Then
[...]
```

1.6.4 Zahlen- und Datumsformate

Die Case-Verzweigung im obigen Beispiel wäre eigentlich überflüssig. Denn um ein Datum in den entsprechenden Wochentag umzuwandeln, steht die Formatfunktion zur Verfügung, die eine Reihe von Optionen zur Verfügung stellt. Die Syntax lautet:

```
Format(Ausdruck[,Format[,firstdayofweek[, firstweekofyear]]])
```

Für die beiden Optionen `firstdayofweek` und `firstdayofyear` stehen die Werte zur Verfügung, die Sie in der unten aufgeführten Tabelle finden. Es können formatiert werden: Zahlen, Zeichenketten, Datums- und Zeitangaben. Wichtig dabei ist, dass der Parameter immer als Text, das heißt mit Anführungszeichen, gekennzeichnet werden muss.

OOo hat einen Fehler in der Funktion `Format`. Leider funktioniert sie nicht für Datums- und Zeitangaben und Texte!

Zahlenformate:

Tabelle 1.12 Zahlenformate

Zeichen	Beschreibung	Beispiel
0	Zeigt die Zahl ohne Formatierung an	Format(1234, "0") liefert 1234, Format(1234, "0.00") liefert 1234,00, Format(1234.5678, "0") liefert 1235 Format(1234.5678, "0.00") liefert 1234,57
#	Platzhalter für eine Ziffer, die nur angezeigt wird, wenn sich an dieser Stelle eine Ziffer befindet. Ge-	Format(1234, "#,##0") liefert 1.234, Format(123, "#,##0") liefert 123

Zeichen	Beschreibung	Beispiel
	dacht für Tausendertrennzeichen.	
%	Multipliziert die Zahl mit 100 und fügt ein %-Zeichen an	Format(0.15, "#,##0.00%") liefert 15,00%, Format(0.125, "0.00%") liefert 12,50%
E- E+ e- e+	Wissenschaftliche Zahlenschreibweise	Format(1250000, "0.00 E+00") liefert 1,25 E+06 Format(1250000, "0.00 E-00") liefert 1,25 E06 Format(1250000, "0.00 e+0") liefert 1,25 e+6 Format(0.125, "0.00 e-0") liefert 1,25 e-1 Format(0.000125, "0.00 E-00") liefert 1,25 E-04
+, -, $, (,), Leerzeichen	Können zur Darstellung direkt in die Formatierung eingefügt werden	Format(1234, "#,##0.00 ") liefert 1.234,00
currency	Liefert das voreingestellte Währungsformat	Format(1234, "currency") liefert 1.234,00 € (wenn so voreingestellt)
\	Das nächste Zeichen wird als Zeichen und nicht als Formatierung ausgegeben. Das „\" verschwindet in der Anzeige.	Format(1234, "#,##0.00 \E\u\r\o") liefert 1.234,00 Euro

Tabelle 1.13 Datums- und Zeitformatierungen:

Zeichen	Beschreibung	Beispiel
/	Datumstrennzeichen	
d	Zeigt den Tag als Zahl	Format("2.9.2005", "d/m/yy") sollte 2.9.05 liefern
dd	Zeigt den Tag als Zahl, wobei Tage zwischen 1 und 9 mit führender Null dargestellt werden	Format("2.9.2005", "dd/mm/yyyy") sollte 02.09.2005 liefern
ddd	Zeigt den Wochentag als Abkürzung	Format("2.9.2005", "ddd") sollte Fr liefern
dddd	Zeigt den Wochentag ausgeschrieben	Format("2 / 9 / 2005", "dddd, \d\e\n dd. mmmm yyyy") sollte Freitag, den 02.09.2005, liefern
w	Der Wochentag als Zahl (1 = Sonntag, 7 = Samstag)	Format("2.9.2005", "w") sollte 6 liefern
ww	Die Kalenderwoche (1 – 54)	Format("2.9.2005", "w") sollte 36 liefern
m	Monat als Zahl (1 – 12)	Format("2.9.2005", "m") sollte 9 liefern
mm	Monat als Zahl. Monate zwischen 1 und 9 mit führender Null	Format("2.9.2005", "w") sollte 09 liefern
mmm	Monat als Text mit drei Buchstaben (Jan-Dez)	Format("2.9.2005", "w") sollte Sep liefern
mmmm	Vollständiger Monatsname	Format("2.9.2005", "w") sollte September liefern
q	Quartal	Format("2.3.2005", "q") sollte 1 liefern
yy	Jahr als zweistellige Zahl (00-99)	Format("2.9.2005", "w") sollte 05 liefern
yyyy	Jahr als vierstellige Zahl (100-9999)	Format("2.9.2005", "w") sollte 2005 liefern

Tabelle 1.14 Die Zeitangaben:

Zeichen	Beschreibung	Beispiel
:	Zeittrennzeichen	
h	Stunde als Zahl ohne führende Null	Format("2:4", "h:m") sollte 2:4 liefern

Zeichen	Beschreibung	Beispiel
hh	Stunde als Zahl mit führender Null	Format("2:4", "hh:mm") sollte 02:04 liefern
n	Die Minute ohne führende Null	Format("15:4", "hh:mm") sollte 15:04 liefern
nn	Die Minute mit führender Null	Format("15:4", "nn") sollte 04 liefern
s	Die Sekunden ohne führende Null	Format("15:4", "s") sollte 0 liefern
ss	Die Sekunden mit führender Null	Format("15:4", "ss") sollte 00 liefern

Tabelle 1.15 Zeichenformatierungen:

Zeichen	Beschreibung	Beispiel
@	Platzhalter für ein Zeichen. Ist die Variable leer, wird ein Leerzeichen ausgegeben.	Format("Sonnenschein", "@") sollte "Sonnenschein" liefern
&	Platzhalter für ein Zeichen. Ist die Variable leer, wird nichts ausgegeben.	Format("Sonnenschein", "&") sollte "Sonnenschein" liefern
!	Auffüllen aller Platzhalter von links nach rechts	Format("Sonnenschein", "!") sollte "Sonnenschein" liefern
<	Zeigt den Text in Kleinbuchstaben	Format("Sonnenschein", "<") sollte "sonnenschein" liefern
>	Zeigt den Text in Großbuchstaben	Format("Sonnenschein", ">") sollte "SONNENSCHEIN" liefern

Das folgende Beispiel funktioniert nicht, da die Funktion `Format` nicht bei Datums- und Zeitangaben arbeitet.

```
Sub GeburtstagWochentag2()
   Dim datGebdatum As Date
   Dim strWochentag1 As String
   Dim strWochentag2 As String

   datGebdatum = InputBox("Wann bist du geboren? ")
```

```
   strWochentag1 = Format(datGebdatum, "dddd")
   strWochentag2 = Format(DateSerial(Year(Date), _
       Month(datGebdatum), Day(datGebdatum)), "dddd")

   MsgBox "Du bist an einem " & strWochentag1 & _
      " geboren.", 64
   If DateSerial(Year(Date), Month(datGebdatum), _
      Day(datGebdatum)) <= Date Then

      MsgBox "Du hattest dieses Jahr an einem " & _
         strWochentag2 & " Geburtstag."
   Else
      MsgBox "Du wirst dieses Jahr an einem " & _
         strWochentag2 & " Geburtstag haben."

   End If
End Sub
```

1.6.5 Funktionen und Unterprogramme

Benutzerdefinierte Funktionen

Basic bietet eine Reihe von Funktionen. Dennoch kann es vorkommen, dass etwas fehlt oder dass Funktionen benötigt werden, auf die häufig zurückgegriffen werden muss.

Der Benutzer gibt einen Pfadnamen an. Nun muss überprüft werden, ob das letzte Zeichen ein / ist. Falls nicht, wird es angehängt. Da diese Option an mehreren Stellen verwendet wird, kann sie als Funktion ausgelagert werden:

```
Sub PfadEingabe
   Dim strPfad As String
   strPfad = InputBox("Bitte einen Pfad eingeben!")
   strPfad = PfadKorrekt(strPfad)
```

```
   MsgBox "Nun lautet der korrekte Pfad: " & strPfad

End Sub

Function PfadKorrekt(strP As String) As String
   If Right(strP, 1) <> "/" Then
      strP = strP & "/"
   End If
   strP = ConvertFromUrl(strP)
   PfadKorrekt = strP
End Function
```

Die Frage und die Antwort sehen wie folgt aus:

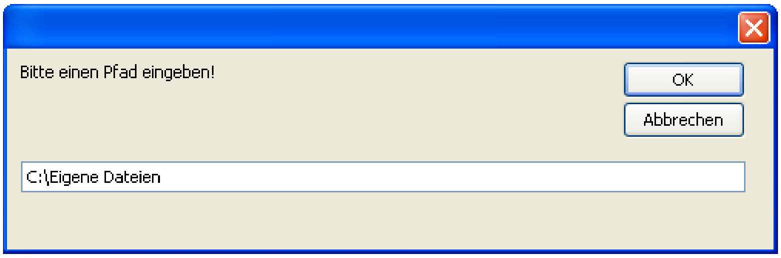


Abbildung 1.26 Eingabe

Abbildung 1.27 Ausgabe

Der Benutzer gibt eine Jahreszahl ein. Von ihr soll überprüft werden, ob das Jahr sich um ein Schaltjahr handelt. Die Regel für die Schaltjahrbestimmung lautet: Ein Jahr ist dann Schaltjahr, wenn es durch 4 teilbar ist. Ist es durch 100 teilbar, dann ist es kein Schaltjahr. Bei 400 ist es allerdings wieder Schaltjahr. Man kann dies über Verzweigungen programmieren.

Dabei stehen mehrere Varianten zur Verfügung:

```
Function Schaltjahr1(intJahreszahl As Integer) As String

   Dim strAusgabe As String

   strAusgabe = " ist kein Schaltjahr."

   If intJahreszahl Mod 4 = 0 Then
      strAusgabe = " ist Schaltjahr."
      If intJahreszahl Mod 100 = 0 Then
         strAusgabe = " ist kein Schaltjahr."
         If intJahreszahl Mod 400 = 0 Then
```

```
            strAusgabe = " ist Schaltjahr."
         End If
      End If
   End If

   Schaltjahr1 = strAusgabe

End Function
```

Man kann in diesem Beispiel auch den umgekehrten Weg gehen:

```
Function Schaltjahr2(intJahreszahl As Integer) As String
   Dim strAusgabe As String

   If intJahreszahl Mod 4 <> 0 Then
      strAusgabe = " ist kein Schaltjahr."
   Else
      If intJahreszahl Mod 100 <> 0 Then
         strAusgabe = " ist Schaltjahr."
       Else
         If intJahreszahl Mod 400 <> 0 Then
            strAusgabe = " ist kein Schaltjahr."
         Else
            strAusgabe = " ist Schaltjahr."
         End If
      End If
   End If

   Schaltjahr2 = strAusgabe
End Function
```

Diese Funktion, die übrigens in OOo-Calc zur Verfügung steht (`IstSchaltjahr`), kann nun von mehreren Stellen aus aufgerufen werden:

```
Sub Eingabe
   Dim intJahreszahl As Integer
```

```
    intJahreszahl = InputBox("Bitte ein gültiges Jahr eingeben!")
    MsgBox intJahreszahl & Schaltjahr1(intJahreszahl)
    MsgBox intJahreszahl & Schaltjahr2(intJahreszahl)
End Sub
```

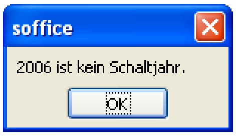


Abbildung 1.28 Das Ergebnis

Es ist unerheblich, ob sich die Funktion oberhalb oder unterhalb der Prozedur befindet. Wichtig sind der Name und die Variablenübergabe! Jedoch legen viele Programmierer Funktionen und Routinen, die von einem Hauptprogramm aufgerufen werden, ans Ende eines Moduls.

Der Aufbau ist der gleiche wie bei Prozeduren: Die Funktion beginnt mit dem Namen `Function`, dem Namen der Funktion, der Liste der übergebenen Variablen. Sie endet mit:

```
End Function
```

Innerhalb der Prozedur oder der Prozeduren, die auf sie zugreifen, steht der Name der Funktion (hier: `PfadKorrekt`) und dahinter die Liste der übergebenen Variablen (hier: `strP`). Basic sucht nun innerhalb des Moduls eine Funktion mit diesem Namen, findet sie und übergibt dort die beiden Variablen (hier: `strPfad`). Mit dieser Variablen wird gearbeitet, das Ergebnis wieder an die Funktion (hier: `PfadKorrekt`) übergeben. Mit diesem Ergebnis wird nun in der ursprünglichen Prozedur weitergearbeitet.

Soll dagegen eine Variable explizit übergeben werden, dann kann die erste Prozedur (A) diese Werte in vielen Programmiersprachen auf zweierlei Weise an Prozedur B übergeben: Werden die alten Werte in der Prozedur noch verwendet, dann geschieht die Übergabe „Call by Value“, das heißt, die alten Werte sind in der Prozedur A noch vorhanden (es werden die Werte direkt übergeben und nicht ein Verweis auf den Speicher). Wird dagegen Prozedur B mit „Call by Reference“ aufgerufen, dann wird der veränderte Werte in A verwendet.

OOo-Basic kennt nur den Übergabetyp „Call by Reference“!

Die Prozedur B könnte so aussehen:

```
Sub B(X As Integer, Y As Integer)
```

```
End Sub
```

In diesem Fall werden die Werte 1 und 2 von A an B übergeben. Dort wird zu ihnen ein Wert addiert – und so werden sie am Ende der Prozedur A angezeigt (als 3 und 4):

```
Sub A
   Dim X As Integer, Y As Integer
   X = 1: Y = 2
   Call B(X, Y)
   MsgBox "x: " & X & " y: " & Y
End Sub

Sub B(X As Integer, Y As Integer)
   X = X + 2: Y = Y + 2
End Sub
```

In OOo-Basic gibt es keine Übergabe „By Value“! Hier müsste man mit einer Hilfsvariablen arbeiten.

Soll eine Funktion modulübergreifend arbeiten, so muss ihr der Modulname mit einem Punkt vorangestellt werden. Damit kann innerhalb eines Projekts von verschiedenen Modulen auf die übergebenen Werte zugegriffen werden.

Leider existiert keine Funktion, die rundet. Sie muss konstruiert werden:

```
Sub Zeige_Wie_du_rundest
   Dim dblZahl As Double, intStellen As Integer
   dblZahl = CDbl(InputBox("Bitte eine Zahl eingeben!"))
   intStellen = InputBox("Auf wie viele Stellen " & _
      "soll gerundet werden?")
   MsgBox "Die Zahl " & dblZahl & " gerundet auf " & _
      intStellen & " Stellen lautet: " & _
      Runden(dblZahl, intStellen)
End Sub

Function Runden(dblX As Double, intStellen As Integer) As Double
```

```
   Dim dblZehnerpotenz As Double
   dblZehnerpotenz = 10 ^ intStellen
   Runden = (Fix((dblX + 0.5 / dblZehnerpotenz) * _
      dblZehnerpotenz)) / _
      dblZehnerpotenz
End Function
```

Abbildung 1.29 Die Rundung

Die Zahl 3,14159 wird auf zwei Stellen nach dem Komma gerundet. Dazu wird die Potenz 10 ^ die Anzahl der Stellen berechnet, also 10 ^ 2 = 100. Die Zahl 0,5 wird nun durch diese Zahl 100 geteilt: 0,005. Diese Zahl wird zu der zu rundenden Zahl addiert und mit 100 multipliziert. Die Nachkommastellen werden abgeschnitten, das Ergebnis wird wieder nach „rechts geschoben“. Hätte die Addition einen „Übertrag“ auf die nächste Ziffer bewirkt, dann wäre auf die nächste Stelle gerundet worden. Wird beispielsweise 3,14159 auf vier Nachkommastellen gerundet, dann ergibt 10 ^ 4 = 1000. 0,5/1000 = 0,0005. 3,14159 + 0,0005 = 3,14164. 3,14164*1000 = 31416,4. Und Fix(31416,4) = 31416. Das wird nun noch durch 1000 geteilt und ergibt: 3,1416.

Allerdings existiert eine weitere Art zu runden: Die Zahl 2,44449 wird mit „Runden“ auf zwei Stellen nach dem Komma auf 2,44 abgerundet und nicht auf 2,45 aufgerundet. Der Grund liegt darin, dass die nächste Stelle nach der zu rundenden kleiner ist als 5, also abgerundet wird. Das kann aber falsch sein. Eine andere Rundungsfunktion sieht folgendermaßen aus:

```
Function Anders_Runden(dblX As Double, intStellen As Integer) As Double
   Dim dblZehnerpotenz As Double
   Dim dblAusgabezahl As Double
   Dim dblSchiebezahl As Double

   dblZehnerpotenz = 10 ^ intStellen

   dblAusgabezahl = Fix(dblX * dblZehnerpotenz) / _
      dblZehnerpotenz
```

```
   dblSchiebezahl = (dblX - dblAusgabezahl) * _
      dblZehnerpotenz * 10

   Anders_Runden = dblAusgabezahl

   Do While dblSchiebezahl >= 4
      If dblSchiebezahl >= 5 Then
         Anders_Runden = dblAusgabezahl + 1 / _
            dblZehnerpotenz
         Exit Function
      Else
         dblSchiebezahl = (dblSchiebezahl - _
            Fix(dblSchiebezahl)) * 10
      End If
   Loop

End Function
```

Die Idee dieser Funktion ist, dass so lange nachgeschaut wird, wie nach der zu rundenden Zahl eine Ziffer gefunden wird, die ein eindeutiges Aufrunden (≥ 5) oder ein eindeutiges Abrunden (< 4) erlaubt. In diesen beiden Fällen wird aufgerundet beziehungsweise abgerundet. Bei der Zahl 4 dagegen wird weitergesucht. Also: Wird die Zahl 2,44449 auf zwei Stellen gerundet, liefert `dblAusgabezahl` den Wert 2,44, `dblSchiebezahl` ist 4,49. Diese Zahl ist zwar größer oder gleich als 4, aber nicht eindeutig größer als 4. Also wird weitergeschoben. Aus 4,49 wird in der Else-Verzweigung ein 4,9, das an `dblSchiebezahl` übergeben wird. Erneut ist die Zahl nicht eindeutig, erneut wird geschoben. Diesmal ist `dblSchiebezahl` 9 und somit klar größer oder gleich 5. Damit kann zur `dblAusgabezahl 1 / dblzehnerpotenz`, also 1 / 100 addiert werden, so dass 2,44449 auf zwei Stellen nach dem Komma auf 2,45 aufgerundet wird.

Abbildung 1.30 Anders_Runden

1.6.6 Unterprogramme: „Makro an Makro: Bitte kommen!“

Ebenso wie Funktionen (siehe Kapitel 1.6.5) andere Programme aufrufen, so können auch Unterprogramme von einem Programm aufgerufen werden. Wenn Sie langen und unübersichtlichen Quellcode haben, so kann dieser dadurch klarer strukturiert werden. Vor allem wenn bestimmte Programmteile mehrmals verwendet werden, sind Unterprogramme zu empfehlen, da sie den Code kürzer und übersichtlicher machen.

Sie können ein Unterprogramm aufrufen, indem Sie `Call` vor den Prozedurnamen setzen oder einfach den Namen selbst als Zeile verwenden:

```
Sub Aufhoeren
   If MsgBox("Sollen wir aufhören?", _
      4 + 32, "Aufhören") = 6 Then Genug
End Sub


Sub Schluss
   If MsgBox("Sollen wir Schluss machen?", _
      4 + 32, "Schluss") = 6 Then Genug
End Sub


Sub Ende
   If MsgBox("Wie wäre es mit einem Ende?", _
      4 + 32, "Ende") = 6 Then Genug
End Sub
Sub Genug
   MsgBox "Verflossen ist das Gold der Tage," & _
   Chr(13) & _
   "Des Abends braun und blaue Farben:" & Chr(13) & _
   "Des Hirten sanfte Flöten starben" & Chr(13) & _
   "Des Abends blau und braune Farben" & Chr(13) & _
   "Verflossen ist das Gold der Tage." & Chr(13) & _
   Chr(13) & Chr(9) & Chr(9) & Chr(9) & "G. Trakl", _
   48, "Rondel"
End Sub
```

Die drei Prozeduren „Aufhoeren“, „Schluss“ und „Ende“ rufen das Unterprogramm „Genug“ auf.

Nach Beendigung der Unterprozedur wird die Hauptprozedur aufgerufen. Da in dieser allerdings kein Befehl zu finden ist, wird sie abgebrochen.

Prinzipiell gibt es keinen bemerkenswerten Unterschied zwischen dem Aufruf einer Funktion und dem Aufruf einer anderen Prozedur. Funktionen können jedoch nur in Verbindung mit Prozeduren arbeiten, aufgerufene Prozeduren könnten auch alleine gestartet werden. Funktionen können Werte über ihren Namen zurückgeben, aufgerufene Prozeduren jedoch nicht. Einige Unterschiede bei Funktionen existieren in Calc Basic. Dort kann man mit Funktionen benutzerdefinierte Funktionen erstellen. Dies wird im Kapitels 1.8 erläutert.

1.6.7 Informationsabfragen

Eine wichtige Funktionskategorie, mit der eingegebene Informationen auf Korrektheit überprüft werden, sind die Informationsabfragen. Vor allem bei Dialogen muss vielerorts getestet werden, ob der Benutzer eine Zahl, ein Text oder ein gültiges Datum eingegeben hat. Dies leisten folgende Funktionen:

Tabelle 1.16 Sämtliche Informationsabfragen

Funktion	Bedeutung	Beispiel
IsDate	überprüft, ob es sich um ein Datum handelt	IsDate(29.2.2005) liefert "False". IsDate(28.2.2005) liefert "True".
IsNumeric	überprüft, ob es sich um eine Zahl handelt	IsNumeric("drei") liefert "False". IsNumeric(3) liefert "True".
IsNull	überprüft, ob eine Variant-Variable leer ist (hier scheint ein Fehler vorzuliegen!)	Dim y As Variant IsNull(y) sollte "True" liefern
IsEmpty	überprüft, ob eine beliebige Variable initialisiert wurde (hier scheint ein Fehler vorzuliegen!)	Dim y As Integer IsEmpty(y) sollte "True" liefern
IsArray	überprüft, ob es sich bei einer Variablen um ein Datenfeld handelt	
IsMissing	überprüft, ob einer Funktion ein optionales Argument übergeben wurde	
IsObject	überprüft, ob es sich um ein Objekt handelt	
IsUnoStruct	überprüft, ob es sich um ein UNO Struct handelt (siehe Kapitel 3)	IsUnoStruct("com.sun.star.beans.Property") liefert "True"

1.7 Schleifen und Sprunganweisungen

Soll ein Befehl mehrmals aufgerufen werden, muss eine Schleife gebaut werden, mit der die einzelnen Schritte mehrmals abgearbeitet werden. Dafür stehen Ihnen zwei Schleifen-typen zur Verfügung:

1.7.1 Die Zählerschleife For ... Next

In der For ... Next-Schleife zählt eine Variable in Einerschritten von einem Startwert zu einem Zielwert hoch. Ist der Endwert erreicht, verlässt das Programm die Schleife und arbeitet die Befehle ab, die nach der For ... Next-Schleife stehen. (Variablen werden in Kapitel 1.4 erläutert.) Die For ... Next-Schleife hat folgende Syntax:

```
For Zähler = Anfangswert To Endwert [Step Schrittweite]
   Anweisungen
Next [Zähler]
```

Dabei ist `Zähler` eine Variable. `Anfangswert` und `Endwert` sind zwei Zahlen (oder Variablen). Würde man die Anweisung

```
For HeineGedichte = 1827 To 1844
```

ausführen lassen, so würde `HeineGedichte` nacheinander die Werte 1827, 1828, 1829, 1830 ... 1844 annehmen. Nach dem ersten Betreten der Schleife, also wenn `HeineGedichte` 1827 ist, werden die folgenden Zeilen abgearbeitet. Bis der Befehl Next auftaucht. Zur besseren Übersicht kann statt `Next` auch

```
Next HeineGedichte
```

stehen.

Angenommen, das Makro soll nicht in Einerschritten hochzählen, sondern in einer anderen Schrittweise, so kann hinter der Zeile:

```
For i = 1 To 322
```

ein `Step` mit einer Zahl folgen. Zum Beispiel:

```
For i = 1 To 322 Step 3
```

Sollte der Endwert überschritten werden, ohne dass der Zielwert genau erreicht wird, so wird die Schleife automatisch beendet. Auch ein Abwärtszählen ist möglich:

```
For i = 322 To 1 Step -3
```

Und sogar ein schrittweises Weiterzählen mit Dezimalzahlen (Achtung: OOo-Basic verlangt die amerikanische Dezimalschreibweise!) ist möglich.

```
For i = 322 To 1 Step -2.5
```

Die For ... Next-Schleife wird folglich entweder gar nicht oder einmal ausgeführt, kann beliebig oft durchlaufen werden und leider auch sogar unendlich oft.

Die folgende Schleife wird genau einmal durchlaufen:

```
For i = 1 To 1
   [Anweisungen]
Next
```

Ist der Startwert dagegen größer als der Anfangswert, dann wird die Schleife nicht durchlaufen. Beispielsweise:

```
For i = 7 To 1
   [Anweisungen]
Next
```

Ein frühzeitiges Verlassen der Schleife wäre mit `Exit For` möglich.

Es existiert noch ein anderer Schleifentyp: die Bedingungsschleife.

1.7.2 Do ... Loop

Die Do ... Loop-Schleife kann kopfgesteuert oder fußgesteuert sein. Die Bedingung kann am Anfang oder am Ende abgefragt werden. Die Syntax der kopfgesteuerten Schleife lautet:

```
Do [While] [Until] Bedingung
   Anweisungen
   [Exit Do]
   Anweisungen
Loop
```

Die Syntax der kopfgesteuerten Schleife lautet:

```
Do
   Anweisungen
   [Exit Do]
   Anweisungen
```

```
Loop [While] [Until] Bedingung
```

Mit `While` oder `Until` wird eine Bedingung überprüft. Ist sie korrekt, wird weitergearbeitet, falls nicht, wird die Schleife verlassen.

Der Unterschied zwischen der Zähler-Schleife und der Bedingungsschleife liegt darin, dass beim ersten Typ die Anzahl der Durchläufe schon feststeht. Ist die Anzahl unklar oder ändert sich innerhalb der Schleife, dann ist einer der folgenden Typen zu verwenden.

Beispiel: Eine Schnecke sitzt vor einer 4,5 Meter hohen Mauer. Jeden Tag klettert sie einen halben Meter nach oben, in jeder Nacht rutscht sie 10% ihrer Gesamthöhe nach unten. Wann ist sie oben?

Also: Sie beginnt bei 0 Meter, klettert am ersten Tag auf 0,5 Meter, rutscht um 0,05 m nach unten, so dass sie den zweiten Tag bei 0,45 m beginnt. Von dort klettert sie auf 0,95 m, um in der folgenden Nacht auf 0,95 m – 0,095 m = 0,855 m zu landen. Dann klettert sie auf 1,355 m und rutscht auf 1,2195 m herunter. Und so weiter. Die Rekursion lässt sich gut in eine Schleife einbauen:

```
Sub Schneckenproblem

   Dim intTage As Integer
   Dim dblMorgenhoehe As Double
   Dim dblAbendhoehe As Double

   dblMorgenhoehe = 0
   dblAbendhoehe = 0
   intTage = 0
   Do While dblAbendhoehe < 4.5
      intTage = intTage + 1
      dblAbendhoehe = dblMorgenhoehe + 0.5
      dblMorgenhoehe = dblAbendhoehe * 0.9
   Loop
   MsgBox "Die Schnecke benötigt " & intTage & " Tage.", 48

End Sub
```

Das Ergebnis soll hier nicht angezeigt werden! Zwei Erläuterungen zu dem Beispiel: Die Zahlen müssen nach der US-amerikanischen Konvention mit „0.9“ und nicht mit „0,9“ eingegeben werden. Wenn 10% abgezogen werden, so kann man statt

```
dblMorgenhoehe = dblAbendhoehe - dblAbendhoehe * 0.1
```

auch schreiben:

```
dblMorgenhoehe = dblAbendhoehe * 0.9
```

denn schließlich interessiert Sie nicht, wie weit die Schnecke herunterrutscht, sondern wo sie landet. Das heißt: Statt 10% abzuziehen, kann er auch die übrigen 90% berechnen. Während des Schleifendurchlaufs zählt ein Zähler (hier: „Tage“) mit, der dann in der Meldungsbox ausgegeben wird. Wann ist die Schnecke oben? Nicht am Ende der Nacht – das heißt, sie rutscht nicht von 5,00 Meter auf 4,50 Meter (denn die Mauer ist schließlich nur 4,50 Meter hoch), sondern sie klettert am Tag von 4,00 Metern auf 4,50 hoch. Und diese Höhe muss abgefangen werden – hier über die Variable „Abendhoehe“. Denn: 5,00 Meter erreicht unsere Schnecke nie!

Vorsicht vor Endlosschleifen. Dann bleibt nur noch der „Ausstieg“ aus dem Makro mit der Tastenkombination <Shift>+<Strg>+<Q>... Die Schnecken lassen grüßen.

Horst wird gefragt, wie alt seine vier Kinder sind. Er gibt als Antwort: „Das Produkt ihrer Alter beträgt 1536, die Summe 30.“ Er überlegt und fügt hinzu: „Die Jüngste heißt Claudia.“ Wie alt sind seine Kinder?

Man kann diese Aufgabe recht einfach durch mehrere ineinander verschachtelte Schleifen lösen. Die erste Schleife läuft vom Produkt 1536 bis 1 herunter. Die zweite Schleife nur noch von der Zahl der ersten Schleife bis 1, die dritte von der Zahl der zweiten Schleife und so weiter. In jeder Schleife wird überprüft, ob der Schleifenzähler ein Teiler von 1536 ist. Falls ja, dann läuft die nächste Schleife los. Im Inneren der vierten Schleife werden die Summe und das Produkt der vier Zahlen gezogen. Beträgt die Summe 30 und das Produkt 1536, dann ist eine Lösung gefunden. Für diese Aufgabe existieren zwei Lösungen: (16/6/4/4) und (12/8/8/2). Da es eine jüngste Tochter gibt, muss die zweite Lösung die korrekte sein. Dies kann übrigens auch in die (innere) Schleife eingebaut werden.

```
Sub KindAlterBerechnen
   Const KPRODUKT = 1536
   Const KSUMME = 30
   Dim dblAlter1 As Double
   Dim dblAlter2 As Double
   Dim dblAlter3 As Double
   Dim dblAlter4 As Double

   For dblAlter1 = KProdukt To 1 Step -1
      If KPRODUKT Mod dblAlter1 = 0 Then
         For dblAlter2 = dblAlter1 To 1 Step -1
```

```
            If (KPRODUKT / dblAlter1) Mod dblAlter2 = 0 Then
               For dblAlter3 = dblAlter2 To 1 Step -1
                  If (KPRODUKT / dblAlter2) Mod dblAlter3 = 0 Then
                     For dblAlter4 = dblAlter3 To 1 Step -1
                        If (KPRODUKT / dblAlter3) Mod _
                        dblAlter4 = 0 Then
                           If (dblAlter1 + dblAlter2 + _
                           dblAlter3 + dblAlter4 = _
                           KSUMME) And (dblAlter1 * dblAlter2 _
                           * dblAlter3 * dblAlter4 = _
                           KPRODUKT) Then
                              MsgBox "Horsts Kinder haben " & _
                                 "folgendes Alter:" _
                                 & Chr(13) & dblAlter1 & "/" & _
                                 dblAlter2 & "/" & dblAlter3 & _
                                 "/" & dblAlter4
                           End If
                        End If
                     Next
                  End If
               Next
            End If
         Next
      End If
   Next
End Sub
```

Man muss nur die innere Schleife modifizieren in:

```
                     For dblAlter4 = dblAlter3  - 1 To 1 Step -1
```

dann wird nur die eine Lösung angezeigt. Natürlich kann man die Lösung noch besser ausfeilen. Die vierte, innere Schleife ist überflüssig, da sich die vierte Zahl als Quotient von KPRODUKT und dem Produkt aus dblAlter1, dblAlter2 und dblAlter3 berechnet. Sicherlich könnte man die Schleifen auch in einer Funktion auslagern. Dies sei dem Leser überlassen.

Sehr viel schneller läuft die Schleife, wenn Sie die Zählung von unten beginnen und nach oben (maximal 30) hochzählen lassen und erst in der vierten Schleife überprüfen, ob die vier Lebensalter die Bedingungen Produkt und Summe erfüllen:

```
For dblAlter1 = 1 To KSumme
   For dblAlter2 = dblAlter1 + 1 To KSumme
      For dblAlter3 = dblAlter2 To 30
         For dblAlter4 = dblAlter3 To 30
            If dblAlter1 * dblAlter2 * dblAlter3 * dblAlter4 = _
               KProdukt And dblAlter1 + dblAlter2 + dblAlter3 + _
               dblAlter4 = 30 Then
                  MsgBox "Horsts Kinder haben " & _
                  "folgendes Alter:" _
                  & Chr(13) & dblAlter1 & "/" & _
                  dblAlter2 & "/" & dblAlter3 & _
                  "/" & dblAlter4
              Exit Sub
              End If
          Next
       Next
   Next
Next
```

Eine Benutzerin oder ein Benutzer wird nach ihrem oder seinem Geschlecht gefragt. Nun soll diese Frage penetrant so lange wiederholt werden, bis sie oder er ein korrektes „w“ oder „m“ eingibt.

```
Sub Begruessung1
   Dim strName As String
   Dim strGeschlecht As String
   Do
   strGeschlecht = InputBox("Wie lautet Ihr Geschlecht?" & _
   Chr(13) & "Bitte ""m"" oder ""w"" eingeben!", "Name")

   If strGeschlecht = "w" Or strGeschlecht = "W" Then
```

```
      strName = InputBox("Wie lautet Ihr Name?", "Name")
      MsgBox "Hallo, liebe " & strName
   ElseIf LCase(strGeschlecht) = "m" Then
      strName = InputBox("Wie lautet Ihr Name?", "Name")
      MsgBox "Hallo, lieber " & strName
   End If
   Loop Until LCase(strGeschlecht) = "w" Or _
      LCase(strGeschlecht) = "m"
End Sub
```

Die Loop-Schleife kann auch anders formuliert werden:

```
Loop While LCase(strGeschlecht) <> "w" And _
   LCase(strGeschlecht) <> "m"
```

Zwar wurde vor einiger Zeit die Fermat'sche These gelöst, aber dennoch gibt es eine Reihe von Problemen, die noch nicht gelöst sind. Das Collatz'sche Problem gehört dazu:

Man nehme eine beliebige Zahl. Ist diese Zahl gerade, dann wird sie durch 2 geteilt. Ist sie dagegen ungerade, dann wird sie mit drei multipliziert und um eins vergrößert; beispielsweise 20:

20 → 10 → 5 → 16 → 8 → 4 → 2 → 1 → 4 → 2 → 1 → 4 → 2 → 1 → ...

oder 21:

21 → 64 → 32 → 16 → 8 → 4 → 2 → 1 → 4 → 2 → 1 → ...

oder 19:

19 → 58 → 29 → 88 → 44 → 22 → 11 → 34 → 17 → 52 → 26 → 13 → 40 → 20 → ...

Jede Reihe endet schließlich in der Folge 1 → 4 → 2 → 1. Schreiben Sie ein Programm, bei dem der Benutzer eine Zahl eingibt, die mit der Collatz'schen Methode auf 1 zurückgeführt wird.

```
Sub CollatzschesProblem
   Dim dblZahl As Double
   Dim strAText As String

   dblZahl = InputBox("Welche Zahl soll getestet werden?")
   strAText = dblZahl
```

```
   Do Until dblZahl = 1
      If dblZahl Mod 2 = 0 Then
         dblZahl = dblZahl / 2
      Else
         dblZahl = dblZahl * 3 + 1
      End If
   strAText = strAText & ", " & dblZahl
   Loop
   MsgBox strAText
End Sub
```

1.7.3 Rekursionen

Eine besondere Art der Schleifen sind Funktionen und Prozeduren, die sich selbst aufrufen. Damit sie sich nicht unendlich oft aufrufen, stehen diese Aufrufe in der Regel in einer Verzweigung, das heißt einer Bedingung.

Das Standardbeispiel für Rekursionen ist das Berechnen der Fakultät. Die Fakultät einer Zahl ist das Produkt aller Zahlen (ab 1) bis zu der Zahl.

Die Fakultät von 5 lautet 120, da

```
5! = 1*2*3*4*5
```

oder allgemein:

```
n! = n*(n-1)*(n-2)* ... * 2 * 1
```

Dies kann über eine Zählerschleife berechnet werden:

```
Sub Fakultaet1
   Dim intZahl As Integer
   Dim dblFakultaet As Double
   Dim intZaehler As Integer
   intZahl = InputBox("Bitte eine Zahl zur Fakultätsberechnung")
   dblFakultaet = 1
   For intZaehler = 1 To intZahl
      dblFakultaet = dblFakultaet * intZaehler
   Next
```

```
    MsgBox "Die Fakultät von " & intZahl & _
       " lautet: " & dblFakultaet
End Sub
```

Oder mit einer Bedingungsschleife:

```
Sub Fakultaet2
   Dim intZahl As Integer
   Dim dblFakultaet As Double

   intZahl = InputBox("Bitte eine Zahl zur Fakultätsberechnung")
   dblFakultaet = 1
   Do While intZahl > 1
      dblFakultaet = dblFakultaet * intZahl
      intZahl = intZahl - 1
   Loop
   MsgBox "Die Fakultät lautet: " & dblFakultaet
End Sub
```

Oder schließlich rekursiv. Die Funktion Fakultät3 wird so lange aufgerufen, bis `intZahl` den Wert 1 erreicht hat (beim Herabzählen).

```
Function Fakultaet3(intZahl As Integer) As Double
   If intZahl > 1 Then
      Fakultaet3 = intZahl * Fakultaet3(intZahl - 1)
   Else
      Fakultaet3 = 1
   End If
End Function
```

Dies kann mit folgender Prozedur aufgerufen werden:

```
Sub FakultaetBerechnen
   Dim intZahl As Integer
   Dim dblFakultaet As Double
   intZahl = InputBox("Bitte eine Zahl zur Fakultätsberechnung")
   dblFakultaet = Fakultaet3(intZahl)
   MsgBox "Die Fakultät von " & intZahl & _
```

```
      " lautet: " & dblFakultaet
End Sub
```

Oder auch mit einer aufsteigenden Zählung:

```
   Dim intZahl As Integer
   Dim intEndZahl As Integer
Function Fakultaet4(intZahl As Integer) As Double

   Fakultaet4 = 1

   If intZahl <= intEndZahl Then
      Fakultaet4 = intZahl * Fakultaet4(intZahl + 1)
   End If
End Function
```

Sie wird aufgerufen:

```
Sub FakultaetBerechnen
   Dim dblFakultaet As Double
   intZahl = InputBox("Bitte eine Zahl zur Fakultätsberechnung")

   intEndZahl = intZahl
   dblFakultaet = Fakultaet4(1)

   MsgBox "Die Fakultät von " & intZahl & _
      " lautet: " & dblFakultaet
End Sub
```

Zugegeben: Die erste Variante ist natürlich eleganter!

Leonardo Fibonacci (1170 – 1250) entdeckte eine Reihe, die mit 1 und 1 beginnt. Jede folgende Zahl entsteht als Summe der beiden Vorgänger.

Also: 1, 1, 2, 3, 5, 8, 13, 21, 34 und so weiter.

Die iterative Lösung stellt sicherlich kein Problem dar. Die iterative Lösung addiert die letzten beiden Werte:

```
Sub Fibonacci_Zahlen1
```

```
   Dim i1 As Double, i2 As Double, i3 As Double
   Dim intAntwort As Integer
   Dim strFibonacci As String

   i1 = 1: i2 = 1
   strFibonacci = "1, 1"

   Do Until intAntwort = 7
      i3 = i1 + i2
      strFibonacci = strFibonacci & ", " & Format(i3, "#,##0")
      i1 = i2: i2 = i3
      intAntwort = MsgBox("Die letzte Fibonacci-Zahl war " & _
         Format(i3, "#,##0") & _
         "." & Chr(13) & "Die Reihe lautet: " & _
         strFibonacci & Chr(13) & Chr(13) & _
         "Möchten Sie eine weitere sehen?", 4)
   Loop
End Sub
```

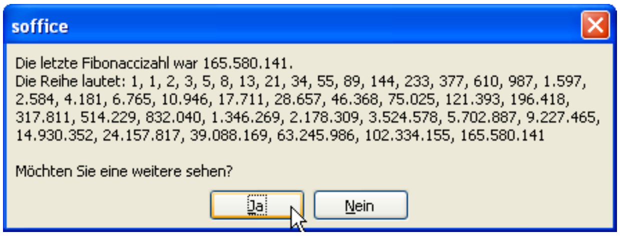


Abbildung 1.31 Fibonacci_Zahlen1

Die rekursive Variante ruft sich selbst auf. Dabei muss man wissen, dass F(x) = F(x-1) + F(x-2). So ergibt sich die rekursive Funktion:

```
Function Fibonacci_Berechnen(dblAnzahl As Double)

   If dblAnzahl < 3 Then
      Fibonacci_Berechnen = 1
```

```
   Else
      Fibonacci_Berechnen = Fibonacci_Berechnen(dblAnzahl - 1) _
      + Fibonacci_Berechnen(dblAnzahl - 2)
   End If
End Function
```

Diese Funktion könnte beispielsweise über folgende Prozedur aufgerufen werden:

```
Sub Fibonacci_Zahlen2
   Dim dblAnzahl As Double

   dblAnzahl = InputBox("Welche Fibonaccizahl wird berechnet?")
   MsgBox "Die " & dblAnzahl & _
      ". Fibonaccizahl lautet: " & _
      Format(Fibonacci_Berechnen(dblAnzahl), "#,##0")
End Sub
```

1.7.4 GoTo (Sprung und Marke)

Es ist sehr unschön, aber es funktioniert! Sie können mit `GoTo` an jede beliebige Stelle im Programm springen. Das heißt: Nach `GoTo` folgt der Name der Sprungmarke. Die Marke selbst steht im Code gefolgt von einem Doppelpunkt am Ende.

```
GoTo Gefängnis
Über Los gehen
Geld (4,000 €) einziehen
Umwege machen
Gefängnis:
Sie begeben sich direkt ins Gefängnis. Sie gehen nicht über Los, Sie ziehen keine
4.000 € ein.
```

Verwenden Sie kein `GoTo`! Diese Sprunganweisung ist sehr unschön. Zwar kann sie praktisch und schnell sein, aber jedes Programm wird unübersichtlich mit GoTo-Befehlen. Früher, in älteren Basic-Versionen, hatte diese Sprunganweisung sicherlich ihre Berechtigung – zu Zeiten, als man in Basic noch Zeilen anspringen konnte. Doch diese sind lange vorbei. Dennoch existiert ein Anwendungsbeispiel für `GoTo`: Nach dem Start einer Prozedur können mögliche Fehler abgefangen werden, indem eine Marke angesprungen wird.

1.8 Eigene Funktionen erstellen

Zwar stellt Calc zirka 350 Funktionen zur Verfügung, jedoch kann es durchaus sein, dass eine Funktion für die tägliche Arbeit fehlt. Wie man solche Funktionen selbst erstellen kann, wird im Folgenden gezeigt. Beispielsweise:

```
Function Busch
   Dim strLaempel1 As String, strLaempel2 As String
   Dim strLaempel3 As String, strLaempel4 As String
   Dim strLaempel5 As String, strLaempel6 As String
   Dim strLaempel7 As String, strLaempel8 As String
   Dim strLaempel9 As String, strLaempel10 As String
   strLaempel1 = "Also lautet ein Beschluss"
   strLaempel2 = "Dass der Mensch was lernen muss"
   strLaempel3 = "Nicht allein das Abc"
   strLaempel4 = "Bringt den Menschen in die Höh"
   strLaempel5 = "Nicht allein in Schreiben, Lesen"
   strLaempel6 = "Übt sich ein vernünftig Wesen"
   strLaempel7 = "Nicht allein in Rechnungssachen"
   strLaempel8 = "Soll der Mensch sich Mühe machen"
   strLaempel9 = "Sondern auch der Weisheit Lehren"
   strLaempel10 = "Muss man mit Vergnügen hören"
   Busch = strLaempel1 & " / " & strLaempel2 & " / " & strLaempel3 & _
   " / " & strLaempel4 & " / " & strLaempel5 & " / " & strLaempel6 & _
   " / " & strLaempel7 & " / " & strLaempel8 & " / " & strLaempel9 & _
   " / " & strLaempel10
End Function
```

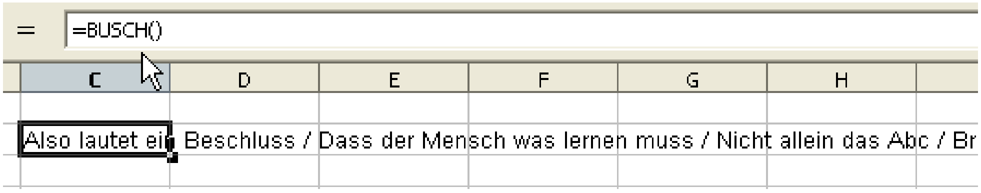


Abbildung 1.32 Die erste Funktion

Der Benutzer gibt die beiden Variablen für a und b ein, mit denen die quadratische Gleichung

0 = x² + a * x + b

berechnet werden soll.

Die Lösung der Gleichung lautet:

$$x_{1,2} = -\left(\frac{a}{2}\right) \pm \sqrt{\left(\frac{a}{2}\right)^2 - b}$$

Sie funktioniert nur, wenn die Diskriminante ist:

$$\left(\frac{a}{2}\right)^2 - b \geq 0$$

Also wird überprüft, ob

$$\left(\frac{a}{2}\right)^2 - b < 0$$

ist. Wenn dies der Fall ist, so existiert keine Lösung. Sollte jedoch

$$\left(\frac{a}{2}\right)^2 - b = 0$$

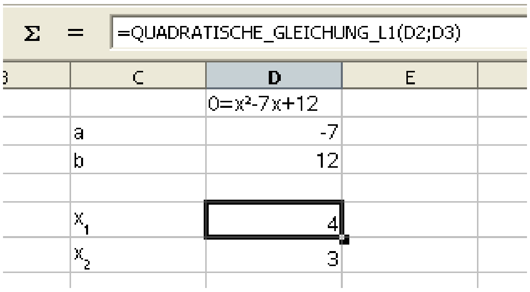
sein, so existiert genau eine Lösung, nämlich: $-\frac{a}{2}$. Andernfalls gibt es die beiden oben angegebenen Lösungen. Der Benutzer wird aufgefordert, die beiden Variablen für a und b auszurechnen. Dann wird überprüft, ob es keine Lösung gibt. Danach wird getestet, ob es nur eine Lösung gibt oder zwei. Diese werden in einem Meldungsfenster angezeigt.

```
Function Quadratische_Gleichung_L1(a As Double, b As Double)
   Dim dblD As Double
   dblD = (a / 2) ^ 2 - b
   If dblD < 0 Then
      Quadratische_Gleichung_L1 = "#Keine Lösung"
   Else
      Quadratische_Gleichung_L1 = -(a / 2) + Sqr(dblD)
   End If
End Function
```

```
Function Quadratische_Gleichung_L2(a As Double, b As Double)
   Dim dblD As Double
   dblD = (a / 2) ^ 2 - b
   If dblD < 0 Then
      Quadratische_Gleichung_L2 = "#Keine Lösung"
   Else
      Quadratische_Gleichung_L2 = -(a / 2) - Sqr(dblD)
   End If
End Function
```

Testen kann man diese Funktionen aus einer anderen Prozedur heraus:

```
   Dim p as Double
   Dim q as Double

   p = InputBox("Bitte Zahl p für x² + px + q eingeben!")
   q = InputBox("Bitte Zahl q für x² + px + q eingeben!")

   MsgBox Quadratische_Gleichung_L1(p, q)
   MsgBox Quadratische_Gleichung_L2(p, q)
```



=QUADRATISCHE_GLEICHUNG_L1(D2;D3)

C	D	E
	0=x²-7x+12	
a	-7	
b	12	
x1	4	
x2	3	

Abbildung 1.33 Zwei Eingaben und zwei Lösungen

Der Benutzer gibt eine Zahl ein, von der überprüft wird, ob es sich um eine Primzahl handelt oder nicht:

```
Function prim(Zahl As Double) As Boolean
   Dim dblZaehler As Double

   If Zahl < 1 Then
      prim = False
      Exit Function
   ElseIf Int(Zahl) <> Zahl Then
      prim = False
      Exit Function
   End If

   For dblZaehler = 2 To Sqr(Zahl) + 1
      If Zahl Mod dblZaehler = 0 Then
         prim = False
         Exit Function
      End If
   Next dblZaehler
   prim = True
End Function
```

Alle Zahlen zwischen 2 und der Wurzel der eingegebenen Zahl werden überprüft, ob sie Teiler der eingegebenen Zahl sind. Wenn sich ein Teiler findet, dann ist die Zahl keine Primzahl – die Berechnung kann beendet werden. Wurden alle Zahlen überprüft und kein Teiler gefunden, so ist die Zahl eine Primzahl. Zugegeben: Die Funktion hat einen kleinen Haken. Zwar könnte jede Zahl bis $1{,}797693 * 10^{324}$ getestet werden, aber die Basic-Funktion `Mod` kann nur Long-Werte verarbeiten. Um den Wertebereich zu vergrößern, müsste die Zeile, in der geprüft wird, korrigiert werden:

```
...
      If Zahl /dblZaehler = Int(Zahl /dblZaehler) Then
...
```

Nun setzt nur Calc die Obergrenze mit 999.999.999.999.990.

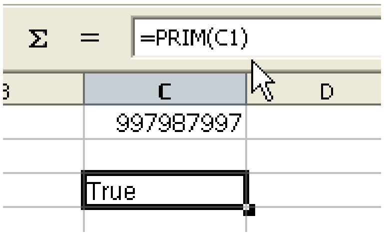


Abbildung 1.34 Der Primzahltest

Oder Sie verwenden eine Bedingungsschleife, die für diese Zwecke besser geeignet ist.

```
Function prim3(Zahl As Double) As Boolean
Dim dblZaehler As Double

   If Zahl < 1 Then
      prim2 = "Fehler"
      Exit Function
   ElseIf Int(Zahl) <> Zahl Then
      prim2 = "Fehler"
      Exit Function
   End If

   dblZaehler = 2
   Do
      If Zahl Mod dblZaehler = 0 Then
         prim2 = False
         Exit Function
      End If
      dblZaehler = dblZaehler + 1
   Loop Until dblZaehler >= Sqr(Zahl) + 1

   prim2 = True
End Function
```

Die Do-Schleife wird betreten, solange der Zähler, der zuvor auf 2 gesetzt wurde, kleiner ist als die Wurzel aus dem eingegebenen Wert. Innerhalb der Schleife wird überprüft, ob der eingegebene

Wert einen Teiler hat. Wenn ja, dann ist die Zahl keine Primzahl. Wenn nein, so wird der Zähler um 1 erhöht, und die Schleife beginnt von vorne mit dem Zählen.

Mit diesem Wissen können Sie nun problemlos den ggT und das kgV berechnen.

Zur Rekapitulation: Folgende Funktionsklassen stehen in Calc zur Verfügung:

1.8.1 Funktionen ohne Parameter

Funktionen, die eine Konstante zurückgeben: in Calc PI(), HEUTE(), JETZT(). Sie sind einfach nachzubilden, wie beispielsweise folgende Funktionen, die aus JavaScript bekannt sind:

```
Function E() As Double
   E = Exp(1)
End Function
Function WURZEL2() As Double
   WURZEL2 = Sqr(2)
End Function

Function WURZEL05() As Double
   WURZEL05 = Sqr(0.5)
End Function
```

Oder auch folgende Konstante, mit welcher der goldene Schnitt berechnet werden kann:

```
Function GOLDENERSCHNITT()
   GOLDENERSCHNITT = 0.5 * Sqr(5) + 0.5
End Function
```

Gerade im Bereich Naturwissenschaften lässt sich eine Vielzahl an Konstanten finden, die als benutzerdefinierte Funktionen gespeichert werden können.

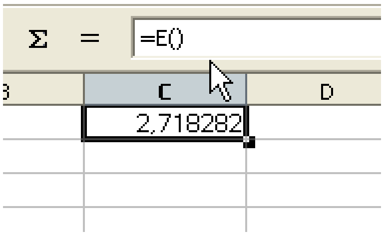


Abbildung 1.35 Die Funktion liefert die Zahl e.

1.8.2 Funktionen, die ein Argument verarbeiten

In Calc sind dies Funktionen wie: WURZEL(Zahl), COS(Zahl), MONAT(Datum), hier einige Beispiele:

```
Function Quersumme(Zahl As Double) As Double
   Dim intZaehler As Integer
   Dim lngQ As Double
   lngQ = 0

   For intZaehler = 1 To Len(CStr(Zahl))
      lngQ = lngQ + CDbl(Mid(CStr(Zahl), intZaehler, 1))
   Next
   Quersumme = lngQ
End Function

Function Endquersumme(Zahl As Double) As Double
   Dim intZaehler As Integer
   Dim lngQ As Double
   lngQ = 0

   For intZaehler = 1 To Len(CStr(zahl))
      lngQ = lngQ + Mid(CStr(zahl), intZaehler, 1)
   Next
   Do While lngQ >= 10
      lngQ = Quersumme(lngQ)
   Loop
   Endquersumme = lngQ
End Function
```

Zu dieser Kategorie zählt auch die oben beschriebene Funktion `prim`.

Diese Funktionen greifen auf eine Zahl, einen Text oder ein Datum zurück und verarbeiten ihn mit Hilfe der Grundrechenarten oder Basic-interner Funktionen. Dabei können Funktionen andere Funktionen verwenden und auch rekursiv rechnen (ENDQUERSUMME).

Vergessen Sie nicht: Laufende Makros können Sie über das Symbol „Anhalten“ beenden oder über die Tastenkombination <Shift>+<Strg>+<Q>.

1.8.3 Funktionen, die mehrere Argumente verarbeiten, die alle eingegeben werden müssen

In Calc zählen dazu: DATUM(Jahr; Monat; Tag), RMZ(Zins; Zzr; Bw), ZÄHLENWENN(Bereich; Suchkriterien), SUCHEN(Suchtext; Text). Ein Beispiel, das in Calc fehlt:

```
Function Suchen_Von_Rechts(Suchtext As String, GesText As String)
   Dim intZaehler As Integer
   For intZaehler = Len(Suchtext) To 1 Step -1
      If mid(LCase(Suchtext), intZaehler, _
      Len(GesText)) = LCase(GesText) Then
         Suchen_Von_Rechts =  intZaehler
         Exit Function
      End If
   Next
   Suchen_Von_Rechts = 0
End Function
```

Es lassen sich noch viele Beispiele ausdenken, in denen mehrere Parameter benötigt werden: finanzmathematische Berechnungen (beispielsweise zur Rentenversicherung), Berechnungen aus naturwissenschaftlichen Diskursen oder auch einfache geometrische Berechnungen. So kann man beispielsweise aus zwei Seiten eines rechtwinkligen Dreiecks die dritte berechnen, aus den drei Seiten eines Dreiecks kann der Flächeninhalt bestimmt werden oder das Volumen eines Kegelstumpfes aus der Höhe und den beiden Radien:

Beispiel: Der Benutzer gibt die Seitenlängen der Katheten eines rechtwinkligen Dreiecks ein und erhält die Länge der Hypothenuse berechnet. Angenommen, diese Berechnung wird häufig verwendet, dann kann das Problem folgendermaßen gelöst werden:

```
Function RechtwinklDreieck(Seite1 As Double, Seite2 As Double) As Double
   RechtwinklDreieck = Sqr(Seite1 ^ 2 + Seite2 ^ 2)
End Function
```

Abbildung 1.36 Die Dreiecksberechnung ($a^2 + b^2 = c^2$)

Angenommen, ein Argument ist optional, dann kann es mit Optional eingefügt werden. Im folgenden Beispiel kann optional ein „u“ (umgangssprachlich) für den Index eingegeben werden:

```
Function Zahlentext(Zahl As Double, _
   Optional Index As String) As String
   Dim intZaehler As Integer
   Dim strZiffer As String
   Dim strZiffertext As String
   Dim strGanzeZahl As String
   For intZaehler = 1 To Len(CStr(Zahl))
      strZiffer = Mid(CStr(Zahl), intZaehler, 1)
      If strZiffer = "0" Then
         strZiffertext = "null"
      ElseIf strZiffer = "1" Then
         strZiffertext = "eins"
      ElseIf strZiffer = "2" Then
         If LCase(Index) = "u" Then
            strZiffertext = "zwo"
         Else
            strZiffertext = "zwei"
         End If
      ElseIf strZiffer = "3" Then
         strZiffertext = "drei"
[...]
      ElseIf strZiffer = "9" Then
         strZiffertext = "neun"
      ElseIf strZiffer = "," Then
```

```
            strZiffertext = "Komma"
        Else
            strZiffertext = Mid(CStr(Zahl), intZaehler, 1)
        End If
        strGanzeZahl = strGanzeZahl & " " & strZiffertext
    Next intZaehler
    Zahlentext = Trim(strGanzeZahl)
End Function
```

Gibt der Benutzer ein „u“ ein, so wird es in der If-Verzweigung verarbeitet, gibt er etwas anderes oder nichts ein, so folgt die Alternative.

1.8.4 Speichern der Basic-Funktionen

Benutzerdefinierte Funktionen können nur dann verwendet werden, wenn die Arbeitsmappe, in der sie gespeichert sind, offen ist. Soll die Funktion für alle Calc-Mappen zur Verfügung stehen, so kann die Datei als normales Calc-Dokument gespeichert werden. Wird sie nun weitergegeben, dann können die Module über Extras | Makros | Makro Verwalten | OOo Basic von einer Datei in eine andere gezogen werden. Wenn Sie dabei die <Strg>-Taste gedrückt halten, dann wird das Modul nicht nur verschoben, sondern auch kopiert. Nun steht es dauerhaft in der Bibliothek „Standard“, das heißt in jeder Calc-Sitzung zur Verfügung.

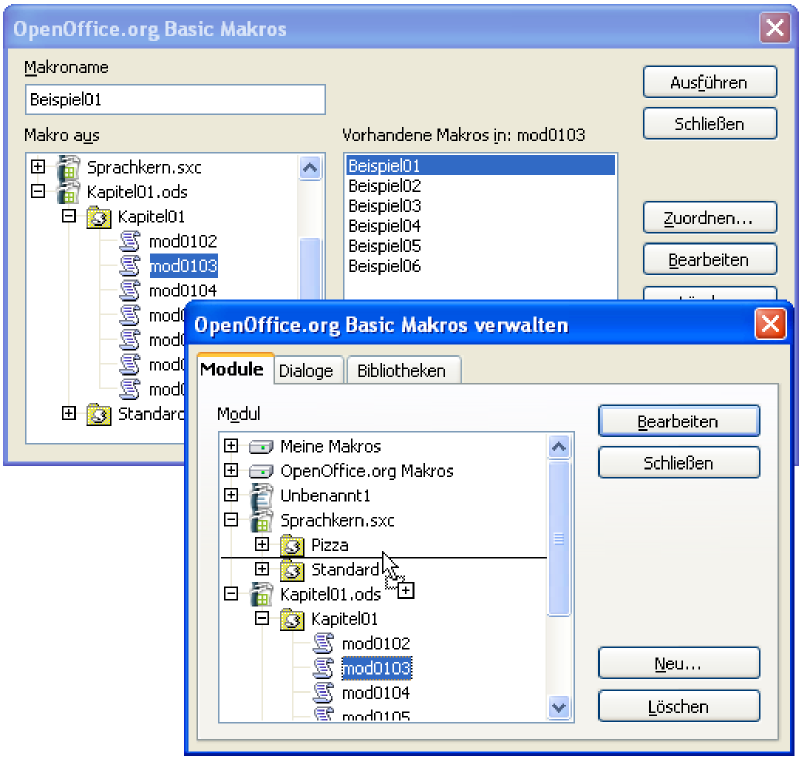


Abbildung 1.37 Das Modul mit den Funktionen wird kopiert.

1.9 Makros starten

Normalerweise wird ein Makro nicht aus der Entwicklungsumgebung, sondern direkt aus der Anwendung heraus gestartet. Dafür stehen verschiedene Möglichkeiten zur Verfügung:

Über den Dialog EXTRAS | MAKROS | MAKRO VERWALTEN | OOO BASIC kann über die Schaltfläche „Zuordnen“ einem Symbol in einem der Anwendungsprogramme ein Makro zugewiesen werden. Analog arbeitet diese Technik für Menüpunkte, Tastenkombinationen und Ereignisse.

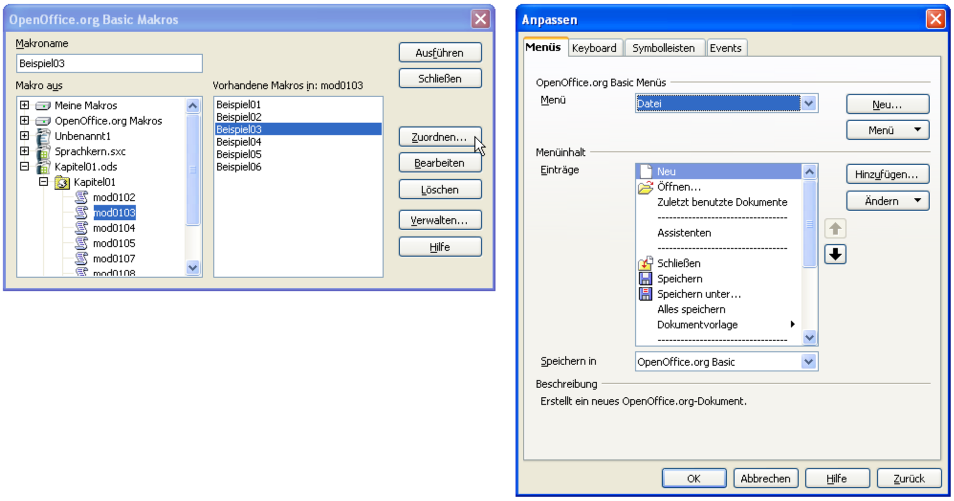


Abbildung 1.38 Makros werden an Symbole, Menüs, Tastenkombinationen und Ereignisse gebunden.

1.10 Dateizugriff

OOo Basic erlaubt den Zugriff auf „sequenzielle Dateien“ und Textdateien. Auch wenn beide Möglichkeiten in der letzten Zeit immer mehr an Bedeutung verlieren, so stellen sie doch interessante Möglichkeit dar, einfache und schnelle Zugriffsmöglichkeiten auf eine überschaubare Menge an Daten zur Verfügung zu stellen.

Andererseits muss oft überprüft werden, ob eine Datei existiert; möglicherweise möchte man eine Datei zur Laufzeit erzeugen und danach wieder löschen. Wie dies funktioniert, wird ebenfalls in diesem Kapitel beschrieben. Dabei stehen die „klassischen“ Themen wie Dateien kopieren, verschieben, löschen, Ordner erstellen und umbenennen im Zentrum. Beginnen wir mit diesem Thema.

1.10.1 Der Zugriff auf Dateien

Noch aus „Urzeiten“ des Betriebssystems DOS stehen Ihnen die bekannten Befehle zur Verfügung:

Tabelle 1.17 Die Elemente der Klasse FileSystem (für Dateien)

Befehl	Bedeutung	Beispiel
GetAttr	bestimmt die Attribute einer Datei	GetAttr("c:\setuplog.txt")
SetAttr	legt Attribute einer Datei fest. Dabei steht: Normal 0 ReadOnly 1 Hidden 2 System 4 Volume 8 Directroy 16 Archive 32	SetAttr("c:\setuplog.txt")
FileDateTime	gibt das Datum und die Zeit der letzten Änderung oder der Erstellung zurück	FileDateTime("c:\setuplog.txt")
FileLen	gibt die Länge einer Datei in Bytes an	FileLen("c:\setuplog.txt")
FileCopy	kopiert eine Datei	FileCopy("c:\setup.txt", "A:\setup.txt")
Name	ändert ein Datei oder einen Verzeichnisnamen	Name "c:\Versuch.txt" As "c:\Test.txt"
Kill	löscht eine Datei	Kill "c:\setuplog.txt"

Tabelle 1.18 Die Elemente der Klasse FileSystem (für Dateien)

Befehl	Bedeutung	Beispiel
CurDir	gibt den aktuellen Pfad an oder legt den aktuellen Pfad fest	CurDir "C" MsgBox CurDir
ChDir	wechselt den aktuellen Ordner oder das aktuelle Laufwerk	ChDir "D:\WINDOWS\SYSTEM"
ChDrive	wechselt das aktuelle Laufwerk	ChDrive "D"
MkDir	erstellt einen neuen Ordner	MkDir "C:\OOo"
RmDir	löscht einen vorhandenen Ordner	RmDir "C:\OOo"
Dir	liefert den Namen einer Datei oder eines Ordners, der in einem Ordner oder auf einem Laufwerk gefunden wird	Dir("C:\OOo") (siehe auch unten)

Für den Befehl `Dir` existiert in der Hilfe ein Beispiel, das leider einige Fehler aufweist. Hier die korrigierte Version:

```
Sub ExampleDir
```

```
   Dim strPath As String
   Dim strDir as String
   Dim strWert As String

   strDir="Verzeichnisse:"
   strPath = "C:" & getPathSeparator
   strWert = Dir$(strPath, 16)
   Do Until strWert = ""
      If strWert <> "." and strWert <> ".." Then
         If (GetAttr( strPath & strWert) AND 16) >0 then
             strDir = strDir & chr(13) & strWert
         End If
      End If
   strWert = Dir$
   Loop

   MsgBox strDir, 0, strPath

End Sub
```

Wird der Wert 16, der für Verzeichnis steht, durch 0 ersetzt, dann können sämtliche Dateien des Ordners ausgelesen werden:

```
Sub ExampleDir2
[...]
   strWert = Dir$(strPath, 0)

   Do Until strWert = ""
      If strWert <> "." and strWert <> ".." Then
         If (GetAttr(strPath & strWert) AND 0) = 0 then
[...]
End Sub
```

Übrigens könnte man sich mit dem Befehl `FileLen` leicht die Dateigröße ausgeben lassen:

```
   strDir = strDir & chr(13) & strWert & "    " & "Länge: " & _
```

```
FileLen(strPath & strWert)
```

1.10.2 Bitweiser Vergleich

Auch wenn es keine große Rolle in OOo-Basic spielt, so soll an dieser Stelle dennoch darauf hingewiesen werden. Im oberen Beispiel wird mit AND ein bitweiser Vergleich durchgeführt.

Tabelle 1.19 Die folgende Tabelle liefert die Datei- und Verzeichnisattribute in dezimaler und Bit-Darstellung

Dezimalwert	Binärwert	Beschreibung	Beschreibung
0	0000 0000	Normal	Bit 0 wurde gesetzt
1	0000 0001	ReadOnly	Bit 1 wurde gesetzt
2	0000 0010	Hidden	Bit 2 wurde gesetzt
4	0000 0100	System	Bit 3 wurde gesetzt
8	0000 1000	Volume	Bit 4 wurde gesetzt
16	0001 0000	Directroy	Bit 5 wurde gesetzt
32	0010 0000	Archive	Bit 6 wurde gesetzt

Wird nun verglichen, ob es sich um ein Verzeichnis handelt, dann werden mit AND die einzelnen Bits verglichen:

ReadOnly	Hidden	System	Volume	Directory	Archive
00 0001 AND 01 0000 00 0000	0000 0010 AND 01 0000 00 0000	0000 0100 AND 01 0000 00 0000	0000 1000 AND 01 0000 00 0000	0001 0000 AND 01 0000 01 0000	0010 0000 AND 01 0000 00 0000

Das Beispiel zeigt deutlich, dass lediglich die Verknüpfung von „Directory“ (0001 0000) AND 19 einen binären Wert > 0 ergibt.

Das obere Beispiel verdeutlich sicherlich auch, dass in einer Bitmaske eine Reihe von unterschiedlichen Informationen gespeichert werden kann, so dass die Datenmenge nicht überflüssig groß wird. Allerdings braucht man heute nicht mehr unbedingt mit Speicherplatz zu geizen …

Es soll überprüft werden, ob auf Laufwerk *C:* ein Ordner *Eigene Dateien* existiert. Falls nicht, dann soll er erstellt werden:

```
Sub OrdnerPruefen
   If Dir("C:\Eigene Dateien", 16) <> "" Then
      MsgBox "Der Ordner ""Eigene Dateien"" existiert."
   Else
      MkDir "C:\Eigene Dateien"
```

```
    MsgBox "Der Ordner ""Eigene Dateien"" wurde nun angelegt."
  End If
End Sub
```

Es soll überprüft werden, ob im Ordner *C:\Eigene Dateien* eine Datei *Makro.ini* existiert. Falls ja, wird das Speicherdatum ermittelt:

```
Sub DateiPruefen
  If Dir("C:\Eigene Dateien\Makro.ini", 0) <> "" Then
    MsgBox "Die Datei wurde erstellt am: " & Chr(13) _
    & FileDateTime("C:\Eigene Dateien\Makro.ini")
  Else
    MsgBox "Die Datei ""Makro.ini"" existiert " & _
    "nicht im Ordner ""Eigene Dateien""."
  End If
End Sub
```

Die vorhandenen Dateien im Ordner *C:\Rechnungen* werden gezählt, und daraus wird eine neue Rechnungsnummer generiert:

```
Sub RechnungsNummerErmitteln
  Dim strDateiname As String
  Const PFAD As String = "C:\Rechnungen\"
  Dim intRechnungsnummer As Integer

  strDateiname = Dir(PFAD)

  Do While strDateiname <> ""
    If strDateiname <> "." And strDateiname <> ".." Then
        If (GetAttr(PFAD & strDateiname)) = 0 Then
         intRechnungsnummer = intRechnungsnummer + 1
      End If
    End If
    strDateiname = Dir
  Loop
  MsgBox Year(Date) & "_" & Format(intRechnungsnummer + 1, "000")
End Sub
```

Mit dem Befehl `Dir` können sowohl die Existenz einer Datei oder eines Verzeichnisses überprüft als auch alle Dateien und Unterordner eines Ordners durchlaufen werden.

Beachten Sie, dass `Dir` unter Windows am Ende des Pfades ein „\“ verlangt.

1.10.3 Informationen in Dateien schreiben und auslesen

Wie der Name „sequenzielle Dateien“ bereits sagt, werden die Daten hintereinander in einer Datei gespeichert. Bevor man mit einer sequenziellen Datei arbeiten kann, muss diese mit

```
Open
```

geöffnet werden. Die vollständige Syntax lautet:

```
n = FreeFile()

Open Name$ For Modus [Access ioMode] [Lock Modus] As #n [Dateilänge]

Close #n
```

Dabei bedeuten:

Tabelle 1.20 Die Elemente des Befehls Open

Begriff	**Erläuterung**
Name$	ist der vollständige Name der zu öffnenden Datei. Also beispielsweise: "C:\Eigene Dateien\Namensliste.txt" oder auch der Name des Gerätetreibers: "LPT1" Die Namen stehen immer in Hochkommata.
Modus	entspricht der Art, wie die Datei geöffnet werden soll: For Input – Die Datei wird zum Lesen geöffnet. For Output – Die Datei wird zum Schreiben geöffnet. For Append – Die Datei wird zum Schreiben geöffnet, aber die neuen Daten überschreiben nicht die alten, sondern hängen sich an sie an. For Binary – Random Access (Access Read, Access Write oder Access Read Write) For Random – Random Access (Access Read, Access Write oder Access Read Write)
Lock Modus	soll verhindert werden, dass andere Benutzer zur gleichen Zeit die Datei verwenden, während sie geöffnet ist, dann stehen folgende Schlüsselwörter zur Verfügung:

Begriff	Erläuterung
	Lock Read — die Datei kann von anderen nicht gelesen werden. Lock Write — in die Datei kann von anderen nicht geschrieben werden. Lock Read Write — in die Datei kann von anderen weder geschrieben noch kann sie gelesen werden.
[#]Dateinummer (n)	ist eine Zahl zwischen 1 und 4. Über diese Zahl kann die geöffnete sequenzielle Datei angesprochen werden.

Wenn Sie eine Datei im Modus `For Input` öffnen, muss die Datei vorhanden sein, sonst kommt es zu einer Fehlermeldung. „For Output“ und „For Append“ verlangen dies nicht: Sie erzeugen eine Datei, falls sie nicht gefunden wird.

Die Datei *Ziel.txt* wird im Modus `For Output` geöffnet. Dabei ist der vollständige Name anzugeben, also zum Beispiel: `C:\Ziel.txt`. Nun werden die neuen Datensätze eingetragen und die alten gelöscht. Die Nummerierung beginnt bei 1. In die erste Zeile wird mit `Write` die Zeile für die Überschrift eingetragen, das heißt die Beschriftung der Spalten. Und mit Hilfe einer Schleife werden die einzelnen Personen und deren Geschlecht eingelesen. Nach Beendigung wird die Datei *Ziel.txt* geschlossen.

```
Sub PersonenEinlesen
   Dim strPersName As String
   Dim strGeschlecht As String
   Dim intAntwort As Integer

   Open "c:\Ziel.txt" For Output As #1
   Write #1, "Name", "Geschlecht"
   intAntwort = 6
   Do While intAntwort = 6
      strPersName = InputBox("Bitte einen Namen eingeben!")
      strGeschlecht = InputBox("Bitte das Geschlecht " & _
         " (""m"" oder ""w"") eingeben!")
      Write #1, strPersName, strGeschlecht
      intAntwort = MsgBox("Möchten Sie noch einen " & _
         "Namen eingeben?", 4)
   Loop
   Close #1
```

```
End Sub
```

Umgekehrt können die Datensatzeinträge wieder ausgelesen werden:

```
Sub PersonenAuslesen
   Dim strPersName As String
   Dim strGeschlecht As String
   Dim i As Integer

   Open "C:\Ziel.txt" For Input As #1
      Do While Not EOF(1)
         Input #1, strPersName, strGeschlecht
         MsgBox strPersName & ", " & strGeschlecht
      Loop

   Close #1
End Sub
```

1.11 Einige weitere Befehle

1.11.1 Shell

Mit dem Befehl `Shell` kann ein fremdes Programm angestoßen werden. Die Syntax sieht folgendermaßen aus:

```
Shell(Pfadname, WindowStyle, Param, bSync)
```

Der Pfadname muss in URL-Schreibweise vorliegen, also entweder so:

```
Shell(("C:\Programme\Adobe\Acrobat%206.0\Acrobat\Acrobat.exe"), 10)
```

oder so:

```
   Shell(ConvertToURL("C:\Programme\Adobe\Acrobat 6.0\Acrobat\Acrobat.exe"), 10)
```

Tabelle 1.21 Für den Parameter WindowStyle liegen folgende Konstanten vor:

Style	Beschreibung
0	Fokus liegt auf dem verborgenen Programm
1	Fokus liegt auf dem Programm, das in Standardgröße geöffnet wird

2	Fokus liegt auf dem minimierten Programm
3	Fokus liegt auf dem maximierten Programm
4	Standardprogrammgröße, jedoch ohne Fokus
6	Minimiertes Programm, jedoch ohne Fokus
10	Volle Bildschirmansicht

Damit können beispielsweise Batch-Dateien angestoßen werden:

```
Shell("C:\Test.bat")
```

Mit dem dritten Parameter können Werte übergeben werden. Wird der vierte Parameter auf „True“ gesetzt, dann wartet er, bis die Applikation gestartet wurde, und arbeitet erst danach weiter.

1.11.2 Declare

Der Befehl `Declare` erlaubt einen Zugriff auf dll-Dateien. Diese Windows-Technologie steht jedoch unter Linux nicht zur Verfügung. Dieses Kapitel soll hier nicht weiter vertieft werden.

1.12 Fehler werden abgefangen

Was passiert, wenn eine Zahl durch 0 geteilt wird. Damit dies nicht passiert, kann diese Zeile übergangen, oder, noch besser, an eine bestimmte Stelle des Programms gesprungen werden.

```
Sub Sprung()
   On Error GoTo ende
   y = 0
   z = 5 / y
   MsgBox z
ende:
End Sub
```

Was passiert? Gar nichts, weil der Fehler (5 / 0) durch die Sprunganweisung abgefangen wird. Als Alternative kann der Fehler auch mit dem Befehl

```
On Error Resume Next
```

abgefangen werden. Im Gegensatz zur Sprungmarke wird die „fehlerhafte“ Zeile

```
z = 5 / y
```

übergangen und direkt dahinter weitergearbeitet. Das bedeutet, dass das Meldungsfenster gezeigt wird, allerdings ohne Inhalt. In einigen, seltenen Fällen eignet sich der allgemeine Befehl

```
On Error Resume Next
```

besser als

```
On Error GoTo ende
```

dazu, Fehler zu übergehen, weil nur eine Zeile übersprungen, nicht aber das gesamte Makro abgebrochen wird. Allerdings kann mit letzterem besser differenziert und überprüft werden, um welchen Fehler es sich handelt.

Das folgende Programm wird nicht ausgeführt, weil vor dem Start festgestellt wird, dass eine Division durch 0 stattfindet:

```
Sub Sprung3
   On Error Resume Next
   z = 5 / 0
   MsgBox z
ende:
End Sub
```

Dass Fehler unterlaufen sind, ist meist sehr schnell an Ergebnissen sichtbar, muss dennoch für verschiedene mögliche Fälle überprüft werden. Für die Zeit der Programmierung empfiehlt es sich, solche deaktivierenden Fehlerbehandlungen auszuschalten, da sonst mögliche Fehler nicht erkannt und reale Fehler übergangen werden.

Leider wird in OOo-Basic nicht auf jeden Fehler reagiert! Erstaunlicherweise läuft das folgende Programm nicht in die Sprungmarke, sondern übergeht den Fehler und meldet zwei Mal (!) den Wert 0.

```
Sub Sprung3
   On Error GoTo ende
   Dim z As Double
   Dim y As Integer
   MsgBox z
   y = "fünf"
   MsgBox z
```

```
    z = 5 / y
    MsgBox z
ende:
End Sub
```

Das schlimmste und vielleicht das wichtigste Kapitel ist die Fehlervermeidung, -suche und -behebung. Lassen Sie uns eine Reihe von Fehlermöglichkeiten durchgehen und Strategien entwickeln, wie man Fehler umgehen kann.

1.12.1 Fehler in der Eingabe

Tippfehler

Gerne wird in mehrgliedrigen Ausdrücken der zweite Teil vergessen, das heißt nach `If` das `End If`, nach `Do` das `Loop`, nach `For Next` und so weiter.

Wenn Sie die erste Zeile geschrieben haben, dann geben Sie gleich danach die zweite Zeile ein, also so:

```
If Variable > 10 Then


End If
```

Vor allem bei verschachtelten Ausdrücken vergessen Sie so nicht den zweiten Teil des Ausdrucks.

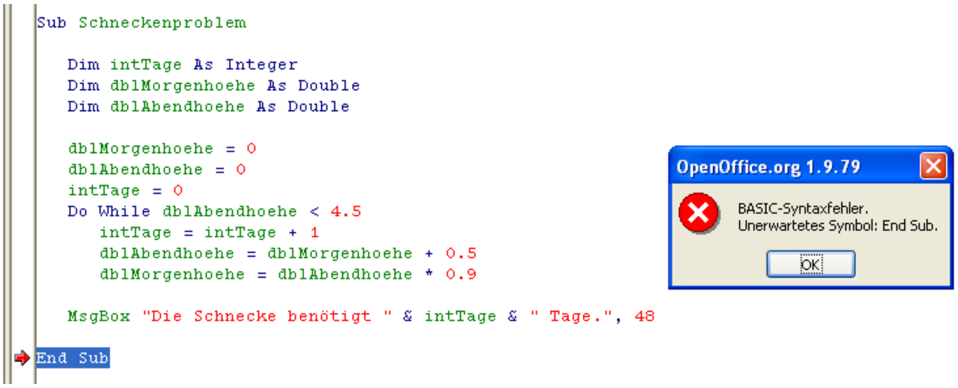


Abbildung 1.39 Ein vergessenes „Then“, „End If“, „Case“ … – hier: ein vergessenes „Loop“

Häufige Tippfehler, die leider nicht angezeigt werden, sind falsche Sprungmarken. Ein

```
GoTo Ziel
```

wird niemals das Ziel erreichen, wenn es

```
Ende:
```

heißt. Auch nicht, wenn es

```
Ziel
```

heißt. Ebenso erhalten Sie eine Fehlermeldung, wenn OOo-Basic zwei Sprungmarken mit dem Namen Ziel findet.

Falsch deklarierte Variablen

Weitere häufige Tippfehler finden sich in Variablen. Deshalb sollten Sie Variablen explizit deklarieren. Überschreiben Sie Ihr Modul mit

```
Option Explicit
```

Es erleichtert die Fehlersuche und vermeidet Fehler.

Nicht nur der Name muss korrekt sein, sondern auch der Typ. Wird er zu klein gewählt, kommt es auch hier zu einer Fehlermeldung:

```
Sub fehler()
   Dim intVariable As Integer
   intVariable = 1234
   intVariable = intVariable * intVariable
   MsgBox intVariable
End Sub
```

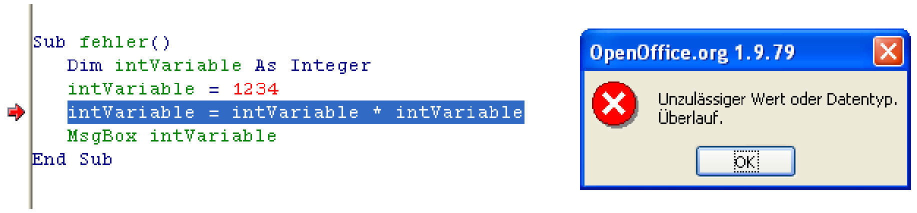


Abbildung 1.40 Falsch deklarierte Variablen

Schlimm sind Schleifen, bei denen ein Zielwert niemals erreicht wird. Dies führt zu einer Endlosschleife. Sollte Ihr Makro jedoch in eine Endlosschleife geraten, kann es mit der Tastenkombination <Shift>+<Strg>+<Q> unterbrochen werden:

```
Sub Endlosschleife
    ' -- kann mit <Strg>+<Shift>+<Q> unterbrochen werden
    Do

    Loop
    MsgBox "Bin ich fertig?"
End Sub
```

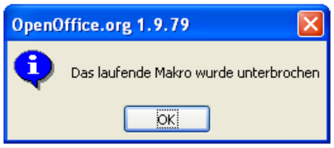


Abbildung 1.41 Endlosschleifen können unterbrochen werden.

1.12.2 Ergebnisse und Zwischenergebnisse

MsgBox

Eine Möglichkeit für Zwischenergebnisse wurde bereits gezeigt: Schreiben Sie in den Code eine Meldungsbox, die ein (oder mehrere) Zwischenergebnisse auswirft. Da das Meldungsfenster ausgeschaltet werden muss, bleibt es so lange am Bildschirm stehen, bis Sie sich von der Richtigkeit der Ergebnisse überzeugt haben. Eine weitere Möglichkeit steht über das Fenster „Beobachter“ zur Verfügung. Dort können Zwischenergebnisse sichtbar gemacht werden. Schreiben oder kopieren Sie Sie den Variablennamen in das Eingabefeld, und bestätigen Sie es, so dass es im unteren Fenster erscheint.

Haltepunkte

Setzen Sie im Code einen Haltepunkt, indem Sie in die linke Leiste doppelklicken, in einer Codezeile <F9> drücken oder indem Sie das Symbol „Haltepunkt“ in der Makroleiste auswählen.

Wenn Sie den Ablauf des Programms angehalten haben, können Sie die Werte der Variablen sehen und danach das Makro weiter laufen lassen. Oder Sie lassen es im Einzelschrittmodus (Symbol „Einzelschritt“ oder <F8>) weiterlaufen. Übrigens wird eine Variable des Beobachters über das Symbol gelöscht.

Das Fenster „Beobachter“

OOo-Basic führt das Programm Zeile für Zeile aus – genauer gesagt: Befehl für Befehl. Stehen mehrere Befehle getrennt mit einem Doppelpunkt in einer Eingabezeile, so wird jeder einzelne Befehl getrennt ausgeführt. Bei einem Befehl, der sich über mehrere Zeilen erstreckt, wird dagegen der ganze Befehl ausgeführt. In welcher Zeile sich die Bearbeitung gerade befindet, wird durch einen gelben Pfeil sichtbar gemacht. Aus der Einzelschrittbearbeitung kann jederzeit das Makro gestartet werden: über den bekannten Weg <F5> oder das Menü EXTRAS | MAKROS | MAKRO oder das entsprechende Symbol:

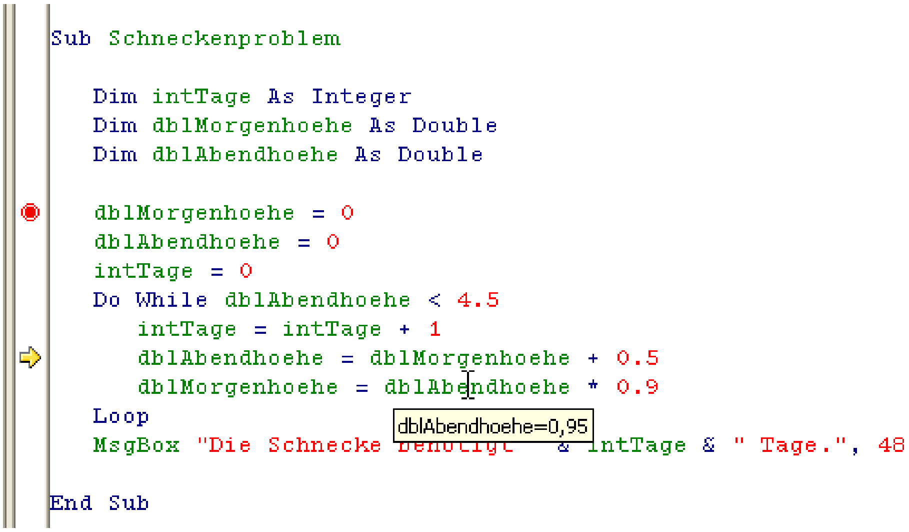


Abbildung 1.42 Der gelbe Pfeil gibt an, in welcher Zeile sich der Einzelschrittmodus befindet.

Das Quick-Info

Um sich die aktuellen Werte während des Einzelschrittablaufs anzuzeigen, genügt es, den Cursor auf eine Variable zu setzen: Der Inhalt wird – ganz gleich, wo innerhalb der Prozedur die Variable steht – in einer QuickInfo-Box angezeigt. So kann auf das Fenster Beobachter und auf das Meldungsfenster als Ausgabemöglichkeiten verzichtet werden.

1.12.3 Und gibt es dennoch Fehler ...

Es gibt jedoch noch andere Fehler. Stellen Sie sich vor, während des Ablaufs eines Makros passiert etwas Unvorhergesehenes. Zum Beispiel soll eine Datei geöffnet werden. Die gibt es aber gar nicht mehr. Oder der Benutzer soll eine Zahl eingeben, der tippt aber statt „10“ „1o“. Damit kann nicht gerechnet werden.

Um solche Fehler abzufangen, steht die Anweisung

```
On Error
```

zur Verfügung. Damit wird geregelt, wie sich die Prozedur bei einem Fehler verhält.

```
On Error Resume Next
```

Mit diesem Aufruf wird die fehlerhafte Zeile übergangen und in der darauf folgenden weitergearbeitet. Diese Anweisung ist dafür gedacht, wenn die Prozedur unbedingt abgearbeitet werden muss. Sollten dann allerdings fehlerhafte Variablen mitgeschleppt werden, so ist davon abzuraten.

Übrigens wird die Fehlernummer an die Variable `Err` übergeben, die abgefragt werden kann. Sollte kein Fehler auftreten, so enthält `Err` den Wert 0:

```
MsgBox Err
```

Mit dieser Fehlerroutine können schwere Fehler, die meist eingeplant sind, abgefangen und zu einer bestimmten Stelle im Programm geleitet werden.

So wie jeder Fehler eine Nummer besitzt (`Err`), hat auch jeder Fehler eine Beschreibung. Diese kann direkt gemeldet werden:

```
MsgBox Error$
```

Die Variable `Erl` liefert die Zeile, in welcher der Fehler aufgetreten ist. Dies kann alles angezeigt werden. Nun kann die Fehlernummer abgefragt und eine gewünschte Aktion ausgeführt werden (Meldung des Fehlers, Speicherung ...). Die Prozedur kann nun beendet oder mit

```
Resume
```

fortgesetzt werden.

Damit geraten Sie in der Regel in eine Endlosschleife, aus der auch die Tastenkombination <Shift>+<Strg>+<Q> nicht mehr heraushilft! Sinnvoller ist meistens, mit

```
Resume Next
```

zu arbeiten. Dann wird eine Zeile nach der fehlerhaften Zeile weitergearbeitet:

```
Sub Nabucco2
   Dim i As Integer
   Dim j As Integer
   On Error GoTo Freiheit

   j = 0
   i = 7 / j
   i = 300000

   Exit Sub
Freiheit:
   MsgBox "Es trat ein Fehler auf" & Chr(13) & _
   Err & ": " & Error$ & Chr(13) & "In Zeile : " & _
   Erl, 16 ,"Fehler aufgetreten"
   Resume Next
```

```
End Sub
```

```
Sub Nabucco2
   Dim i As Integer
   Dim j As Integer
   On Error GoTo Freiheit

   j = 0
   i = 7 / j
   i = 300000
   Exit Sub
Freiheit:

   MsgBox "Es trat ein Fehler auf" & Chr(13) & _
   Err & ": " & Error$ & Chr(13) & "In Zeile : " & _
   Erl, 16 ,"Fehler aufgetreten"

   Resume Next

End Sub
```

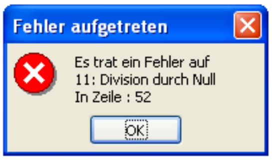


Abbildung 1.43 Der erste der beiden Fehler

Der Name der Sprungmarke ist beliebig, darf natürlich nicht mit einem Schlüsselbegriff oder einer Variablen übereinstimmen. Die Sprungmarke selbst ist nun die Wiederholung des Namens an einer bestimmten Stelle, wobei hinter dem Namen ein Doppelpunkt steht. Fehlersprungmarken müssen sich innerhalb der Prozedur befinden und stehen meistens am Ende des Makros. Damit die Routine nicht dort hineinläuft, kann sie mit

```
Exit Sub
```

oder

```
Exit Function
```

umgangen werden.

```
Sub Nabucco1
On Error GoTo Freiheit
   [Anweisungen...]
   Exit Sub
Freiheit:
   [Anweisungen]
End Sub
```

Soll nach Behandlung der Fehlerroutine wieder zurückgekehrt und das Makro weiter ausgeführt werden, so kann mit `Resume` gearbeitet werden. On Error wirkt sich allerdings nur auf die Hauptprozedur aus, nicht auf mögliche Fehler innerhalb der Fehlerroutine. Sollen diese ebenfalls abgefangen werden, so ist nach der Sprungmarke eine weitere Anweisung

```
On Error Resume Next
```

oder

```
On Error GoTo Marke2
```

anzugeben. Nur so ist gewährleistet, dass mögliche Fehler das Programm nicht durch eine weitere Meldung unterbrechen. Übrigens springt der Befehl

```
Resume
```

zu der Stelle zurück, von der die Fehlerroutine aufgerufen wurde, und führt die gleiche Zeile noch einmal aus. Dies führt zu einem erneuten Fehler und einem Sprung in die Fehlerroutine – eine Endlosschleife also. Vermeiden Sie sie, indem Sie mit

```
Resume Next
```

zurückspringen. Damit wird das Programm ab der nächsten Zeile weiterbearbeitet. Alternativ könnte mit

```
Resume Marke3
```

eine weitere Sprungmarke angesprungen werden, was aber den Code nicht gerade übersichtlich macht.

Sicherlich ist es Ihnen schon aufgefallen, dass in OOo-Basic einige fehlerhafte Zeilen „kommentarlos“ übergangen werden:

```
Sub Nabucco3
   Dim k As Integer

    On Error GoTo Freiheit
   k = "Casablanca"

   MsgBox "Das Programm läuft"
   Exit Sub
Freiheit:

   MsgBox "Es trat ein Fehler auf" & Chr(13) & _
   Err & ": " & Error$ & Chr(13) & "In Zeile : " & Erl, 16 ,"Fehler aufgetreten"

End Sub
```

Andererseits werden bereits schon beim Starten Fehler gemeldet – noch bevor überhaupt eine Zeile ausgeführt wurde:

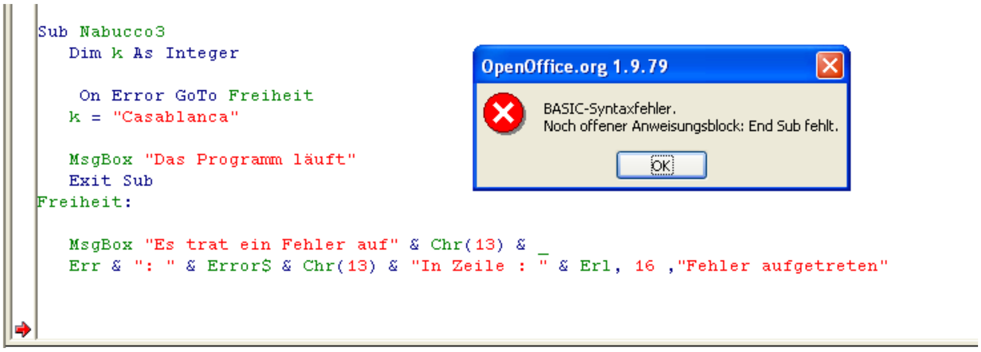


Abbildung 1.44 Diese Prozedur kann nicht gestartet werden.

Neben `Exit Sub` stehen Ihnen noch die beiden Befehle `Wait` und `Stop` zur Verfügung. Mit dem ersten kann eine Anzahl Millisekunden eingegeben werden, mit dem zweiten wird das Programm beendet:

```
Wait(5000)
```

Die Entwicklungsumgebung und der Sprachkern

Damit der Benutzer bequem seine Eingaben tätigen kann, mit Hilfe eines Assistenten Schritt für Schritt durch eine Aufforderungsanleitung geführt wird oder auch Ergebnisse von Auswertungen in einer eleganten Form sehen kann, stellt OOo-Basic Dialoge zur Verfügung. Sie werden auch Formulare, Userforms, Eingabemasken … genannt.

2 Dialoge

2.1 Ein neuer Dialog

Ein neuer Dialog wird wie ein Modul über EXTRAS | MAKROS | DIALOGE ORGANISIEREN (ODER: MAKROS ORGANISIEREN | OOO-BASIC | ORGANIZER | DIALOGE | NEUER DIALOG (oder in OOo 1.1: EXTRAS | MAKROS | MAKRO | VERWALTEN | NEUER DIALOG) erstellt oder über das Kontextmenü des Registerblattes. Dort wird ein neues Modulblatt mit einem Dialog erzeugt.

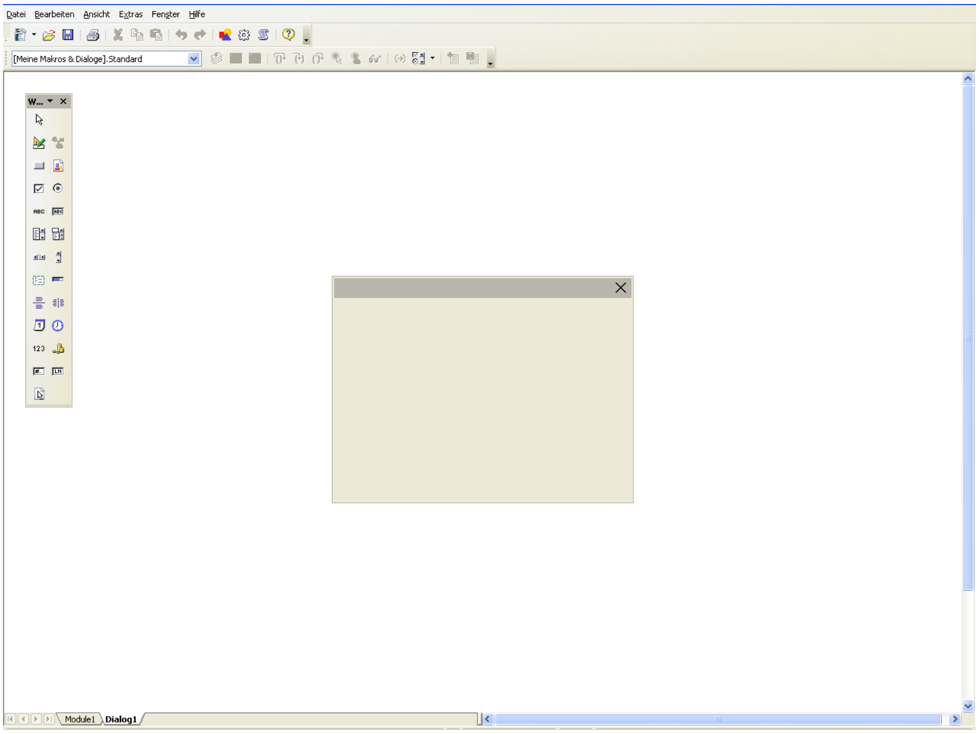


Abbildung 2.45 Ein neuer Dialog

Ebenso wie Module werden auch Dialoge in Bibliotheken verwaltet – für größere Projekte empfiehlt es sich, zuerst eine neue Bibliothek anzulegen und anschließend darin einen Dialog zu erstellen.

Es gibt keine Möglichkeit, einen einzigen Dialog zu exportieren oder von einer Bibliothek in eine andere zu kopieren. Lediglich ganze Bibliotheken können von einer Datei in eine andere Datei hinzugefügt werden. Sie sollten sich also vor dem Erstellen eines Dialogs überlegen, in welche Bibliothek Sie ihn einfügen.

Der Name des Registerblattes, auf dem sich der Dialogs befindet, ist gleich dem Namen des Dialogs, über den er später gesteuert wird.

Der Dialog wird über ein Makro aus einem Modul aufgerufen:

```
Sub Main

   Dim Dlg As Object

   DialogLibraries.LoadLibrary("Standard")
   Dlg = CreateUnoDialog(DialogLibraries.Standard.dlgKaffeeautomat)
   Dlg.Execute()

End Sub
```

„Standard“ ist dabei der Name der Bibliothek, „Kaffeeautomat“ der selbst gewählte Dialogname. Wenn sichergestellt ist, dass die Bibliothek „Standard“ geladen ist, dann kann auf die Zeile

```
   DialogLibraries.LoadLibrary("Standard")
```

verzichtet werden. Der Dialog kann über das Schließen-Symbol oder über die Taste <Abbrechen> geschlossen werden.

Wenn Sie auf einen der vier Ränder klicken, erscheinen die acht Markierungspunkte. An ihnen können die Größe und auch die Startposition des Dialogs verändert werden. Ebenso kann um den Dialog ein großer Rahmen gezogen werden – dann ist er ebenfalls ausgewählt.

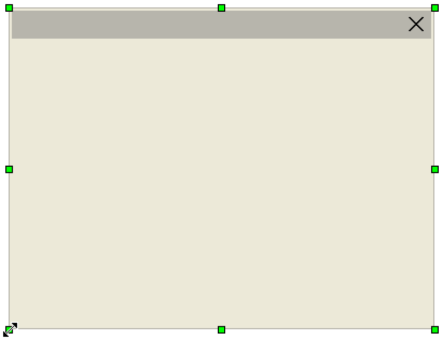


Abbildung 2.46 Der Dialog kann vergrößert werden.

Über das Kontextmenü können nun die Eigenschaften eingestellt werden. Beachten Sie, dass man zwar über den Namen, der in den Eigenschaften geändert werden kann, auf den Dialog zugreifen kann – gestartet wird er jedoch über den Namen, der im Registerblatt angegeben ist. Im Feld „Titel“ kann der Text festgelegt werden, der später in der Titelzeile steht, die Höhe und Breite können numerisch festgelegt werden, ebenso wie die Hintergrundfarbe.

Tabelle 2.22 Einige Eigenschaften des Dialogs:

Eigenschaft	Wert	Beschreibung
Title	String	der Text in der Titelzeile
Enable Model.Enabled	Boolean	der Dialog empfängt keine Benutzereingaben. Achtung: Er kann auch nicht mehr geschlossen werden!
Visible	Boolean	sichtbar
ImplementationName	String	stardiv.Toolkit.UnoDialogControl

Tabelle 2.23 Einige Methoden des Dialogs:

Methode	zurückgegebener Wert	Beschreibung
setTitle(String)	Void	setzt die Titelzeile
getTitle()	String	liefert den Text der Titelzeile
execute()	Integer	startet den Dialog
endExecute()	Void	beendet den Dialog

Vielleicht überrascht es Sie, dass die Methode `execute()` einen Integer-Wert zurückgibt. Mit seiner Hilfe kann herausgefunden werden, wie der Dialog geschlossen wurde. Der Wert 0 wird dann zurückgegeben, wenn der Benutzer <esc> drückt, die Abbrechen-Schaltfläche anklickt oder über das „x“ in der Titelzeile den Dialog schließt. Ebenso wenn der Dialog per Code mit der Methode `endExecute()` geschlossen wurde. Lediglich die OK-Schaltfläche, an der kein Code gebunden ist, liefert den Wert 1 zurück.

2.2 Kontrollfelder (Steuerelemente, Controls)

Um nun Kontrollfelder auf den Dialog zu platzieren, muss die Symbolleiste „Kontrollfelder“ in OOo 1.4 über das entsprechende Icon geöffnet werden (hierfür gibt es – ebenso wie für die Eigenschaften – keinen Menüpunkt), in OOo 2.0 erscheint die „Werkzeugleiste“ automatisch. Auf dem Dialog können die entsprechenden Steuerelemente erzeugt werden, indem ein Feld in der Symbolleiste markiert und auf dem Dialog aufgezogen wird.

2.2.1 Schaltflächen

Am unteren Rand des Dialogs werden zwei Schaltflächen für den OK- und den Abbrechen-Button aufgezogen. Um ein Kontrollfeld bequem markieren zu können, wählen Sie in der Symbolleiste den Auswahlpfeil aus. Denn, falls Sie eines der Kontrollfelder aktiviert haben und ein Rechteck auf dem Dialog aufziehen, dann wird ein neues Steuerelement erzeugt.

Mehrere Kontrollfelder können markiert werden, indem Sie mit der Maus ein Rechteck um sie herum ziehen oder indem sie mit gedrückter <Shift>-Taste einzeln ausgewählt werden. Bei unterschiedlichen Kontrollfeldern erscheinen nun in den Eigenschaften nur diejenigen, über die alle selektierten verfügen.

Um die Beschriftung eines Objektes zu ändern, werden die Eigenschaften benötigt. Dafür gibt es neben dem Kontextmenü auch ein Symbol in der oberen rechten (in OOo 1.4 in der unteren linken) Ecke der Kontrollfelder-Palette.

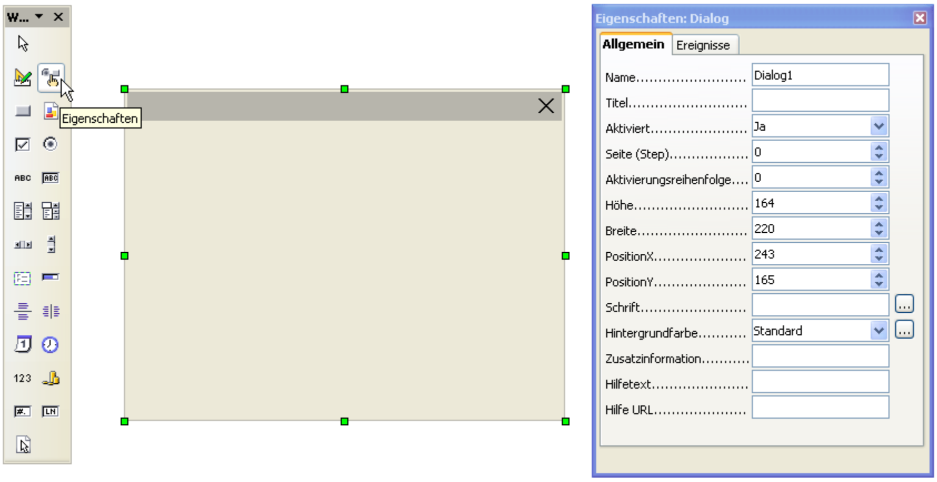


Abbildung 2.47 Die Werkzeug-Palette und die Leiste der Eigenschaften

Die wichtigste Eigenschaft jedes Kontrollfeldes ist sicherlich der Name, da über ihn das Kontrollfeld angesprochen wird.

Da über den „Namen" das Kontrollfeld angesprochen wird, darf natürlich kein Name zwei Mal vergeben werden. Wenn Sie es dennoch (aus Versehen) tun, dann erhalten Sie keine Fehlermeldung, sondern der Name wird stillschweigend zurückgesetzt!

Mit „Titel" ist die Beschriftung gemeint – der Text, der dem Benutzer auf dem Kontrollfeld angezeigt wird. Damit beide Felder bündig ausgerichtet sind, kann über die Eigenschaft „Y-Position" der gleiche Wert eingestellt werden (leider existiert kein Menüpunkt FORMAT | AUSRICHTEN oder Ähnliches). Ebenso kann eine gleiche Größe der Kontrollfelder über „Breite" und „Höhe" festgelegt werden. Über das Icon „Testmodus aktivieren" kann eine Vorschau auf den Dialog erstellt werden.

Damit der Dialog über das Abbrechen- und OK-Kontrollfeld geschlossen wird, kann in den Eigenschaften die „Art der Schaltfläche" auf „Abbrechen" beziehungsweise „OK" eingestellt werden.

Soll über die OK-Schaltfläche jedoch ein Ereignis ausgelöst werden, dann darf die „Art der Schaltfläche" nicht auf „OK" gestellt werden. Dafür findet sich in den Eigenschaften im zweiten Registerblatt „Ereignisse" das Ereignis „Beim Auslösen". Es tritt in Kraft, wenn der Benutzer mit der Maus auf die Schaltfläche klickt oder mit Hilfe der Tabulatortaste den Focus auf diese Schaltfläche setzt und mit <Enter> bestätigt.

In einem Modul wird eine Prozedur erstellt:

```
Sub cmd_OK
   MsgBox "Nun gibt es Kaffee"
End Sub
```

Damit sie aufgerufen werden kann, wird an das Ereignis „Beim Auslösen“ das Makro gebunden. Soll der Dialog danach geschlossen werden, dann kann dies über die Methode

```
Dlg.endExecute()
```

durchgeführt werden. Dann muss jedoch die Objektvariable `Dlg` global im Modul „Start“ deklariert werden:

```
Public Dlg As Object
```

Sie müssen das Makro in den „Ereignissen“ des Eigenschaftendialogs zuerst „zuweisen“ – die alleinige Bestätigung mit „OK“ genügt nicht!

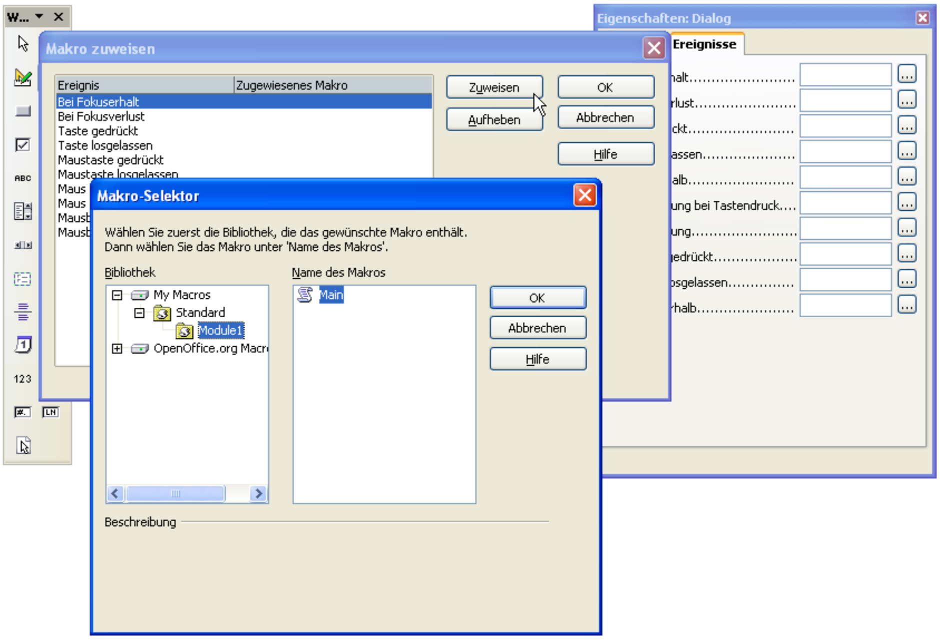


Abbildung 2.48 Einem Ereignis wird ein Makro zugewiesen

Beachten Sie, dass mit dem Ereignis auch der Bibliotheksname und der Modulname an das Kontrollelement gebunden werden. Ein nachträgliches Umbenennen führt dazu, dass Sie erneut das Ereignismakro an das Kontrollelement binden müssen.

Beachten Sie, dass der Testmodus lediglich eine Vorschau darstellt! Die Makros, die an die Kontrollfelder gebunden sind, werden über den Testmodus nicht ausgelöst! Zum Testen muss der Dialog über das Startmakro aufgerufen werden.

Der Titel des Dialogs kann – statt über die Eigenschaften – ebenso über die Property „Title“ per Programmierung gesetzt werden:

```
Dlg.Title = "Kaffeeautomat"
```

2.2.2 Textfelder und Beschriftungsfelder

Auf dem Dialog wird ein Textfeld `txtName` aufgezogen. Damit der Benutzer weiß, was er in dieses Feld schreiben muss, wird ein Beschriftungsfeld `lblName` verwendet. Alternativ (oder zusätzlich) könnte in den Eigenschaften ein Hilfetext festgelegt werden – er wird als Quick-Info beim Darüberfahren mit der Maus angezeigt.

Soll der Cursor beim Start des Dialogs in dem Textfeld sitzen, so ist die Nummer der „Reihenfolge“ vor die Schaltfläche zu setzen, beispielsweise kann das Beschriftungsfeld die Nummer 0 erhalten, das Textfeld die 1.

Lassen Sie sich nicht von der weißen Hintergrundfarbe des Labels in der Entwurfsansicht irritieren – wenn als Hintergrundfarbe „Standard“ gewählt wurde, dann erscheint die Farbe bei der Anzeige nicht.

Mit der Methode `getControl()` wird auf das Steuerelement zugegriffen, mit der Methode `getText()` wird der Beschriftungstext ausgelesen:

```
Sub cmd_OK
   If Dlg.getControl("txtName").getText() = "" Then
      MsgBox "Bitte autorisieren Sie sich!"
   Else
      MsgBox "Hallo " & Dlg.getControl("txtName").getText() & _
         Chr(13) & "Nun gibt es Kaffee"
      Dlg.endExecute()
   End If
End Sub
```

Beachten Sie, dass bei den Steuerelementen zwischen Groß- und Kleinschreibung unterschieden wird!

Tabelle 2.24 Einige Eigenschaften der Befehlsschaltflächen, Labels und Textfelder

Eigenschaft	**Wert**	**Beschreibung**
Name	String	Steuerelementname
Model.Height, Model.Width, Model.PositionX und Model.PositionY	Long	Höhe, Breite, X- und Y-Position des Steuerelementes (in ma-Einheiten)
Enable Model.Enabled	Boolean	das Steuerelement ist aktivierbar
Visible	Boolean	sichtbar
Model.Tabstop	Boolean	ermöglicht die Erreichbarkeit des Steuerelementes über die Tabulatortaste
Model.TabIndex	Long	Position des Steuerelementes in der Aktivierungsreihenfolge
Label Model.Label	String	Beschriftung der Schaltfläche
Text	String	Beschriftung des Labels oder Inhalt des Textfeldes
ImplementationName	String	stardiv.Toolkit.UnoButtonControl (Schaltfläche), stardiv.Toolkit.UnoFixedTextControl (Label) und stardiv.Toolkit.UnoEditControl (Textfeld)

Tabelle 2.25 Einige Methoden der Befehlsschaltflächen, Labels und Textfelder

Methode	**Wert**	**Beschreibung**
setLabel(String)	Void	ändert die Beschriftung bei Schaltflächen
setText(String)	Void	ändert den Text bei Labels und Textfeldern
getText()	String	liefert den Text bei Labels und Textfeldern
setVisible(Boolean)	Void	macht sichtbar oder unsichtbar
setEnable(Boolean)	Void	aktiviert oder schaltet die Aktivierbarkeit aus
setFocus()	Void	setzt den Fokus (Cursor)

Um eine möglichst hohe Plattformunabhängigkeit zu gewährleisten, verwendet OOo zur Angabe von Positionen und Größen innerhalb von Dialogen die interne Einheit namens Map AppFont (ma). Ein ma entspricht einem Achtel der durchschnittlichen Höhe eines Zeichens aus dem im Betriebssystem definierten Systemfonts sowie einem Viertel der Breite eines entsprechenden Zeichens. Durch die Verwendung von ma-Einheiten stellt OOo sicher, dass ein Dialog trotz unterschiedlichen Systemen und bei verschiedenen Systemeinstellungen stets gleich erscheint.

Soll die Größe oder Position von Steuerelementen zur Laufzeit geändert werden, so empfiehlt es sich, die Gesamtgröße des Dialoges zu ermitteln und die Werte für die Steuerelemente an den entsprechenden Teilverhältnissen auszurichten.

Damit nach der Fehlermeldung der Cursor wieder im Textfeld sitzt, kann dies realisiert werden über:

```
MsgBox "Bitte autorisieren Sie sich!"
Dlg.setFocus()
ctlTxt.setFocus()
```

Es funktioniert auch ohne die Zeile:

```
Dlg.setFocus()
```

Schließlich wird noch eine Zahl eingegeben – der Betrag, den der Benutzer zu zahlen hat. Dafür kann ein Währungsfeld verwendet werden. In den Eigenschaften werden die Unter- und Obergrenze festgelegt („Min. Wert“ und „Max. Wert“), das Rundungsintervall, in dem zu große oder zu kleine Beträge gerundet werden („Intervall“), die Anzahl der Nachkommastellen, auf die kaufmännisch gerundet wird („Nachkommastellen“), „Tausendertrennzeichen“ und das „Währungssymbol“. Diese Eigenschaften werden erst beim Verlassen des Feldes wirksam, nicht bei der Eingabe. Das bedeutet, dass eine falsch eingegebene Zahl „1ooo“ erst beim Verlassen auf „0,00 €“ zurückgesetzt wird. Beachten Sie, dass der Wert der Zahlenfelder über die Eigenschaft `Value` abgefragt und gesetzt wird.

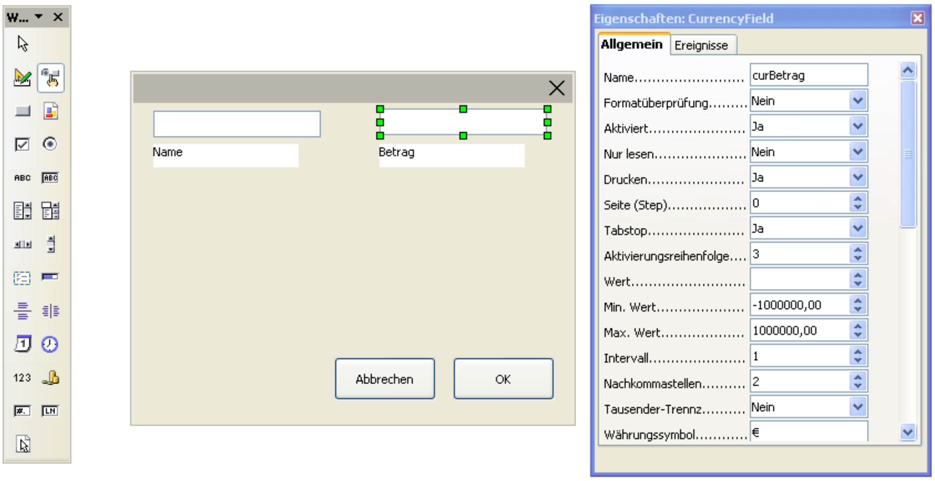


Abbildung 2.49 Textfelder und Beschriftungsfelder

2.2.3 Optionsfelder (Radiobuttons)

Rein zur optischen Gliederung wird auf dem Dialog ein Rahmen aufgezogen. In diesen Dialog werden nun mehrere Optionsfelder eingefügt, von denen eines den Status „ausgewählt“ erhält. Nun kann, wenn die Reihenfolge der Radiobuttons unmittelbar aufeinander folgt, nur einer der Buttons ausgewählt werden. Mit der Eigenschaft „State“ wird abgefragt, welcher der Buttons ausgewählt wurde:

```
If Dlg.getControl("optKaffee").State = True Then
   strGetraenk = "Kaffee"
ElseIf Dlg.getControl("optTee").State = True Then
   strGetraenk = "Tee"
ElseIf Dlg.getControl("optCola").State = True Then
   strGetraenk = "Cola"
ElseIf Dlg.getControl("optBruehe").State = True Then
   strGetraenk = "Brühe"
End If

MsgBox "Hallo " & ctlTxt.getText() & Chr(13) & _
```

```
        "Nun gibt es " & strGetraenk
```

Damit noch eine weitere Gruppe von Optionsfeldern eingefügt werden kann, darf der nächste Radiobutton nicht in der Reihenfolge nach dem letzten der vorhergehenden Gruppe stehen. Eine Lösung sieht einen Rahmen vor, dem eine Reihenfolgenposition vergeben werden kann, eine andere einen Trennstrich.

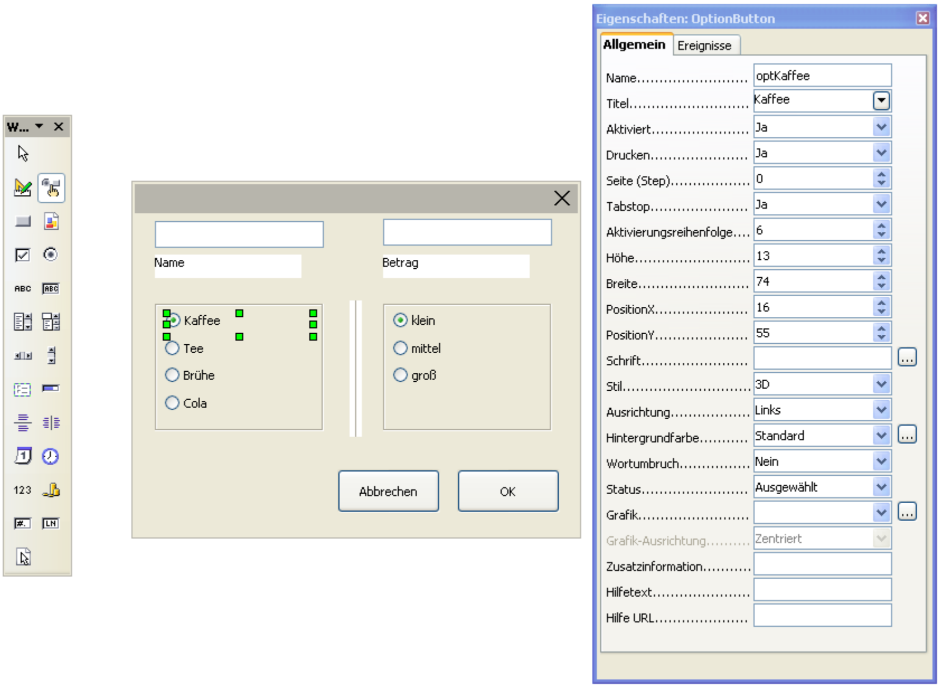


Abbildung 2.50 Optionsfelder

Um den Code etwas schlanker zu halten, könnte ebenso für jeden Radiobutton eine Prozedur erstellt werden:

```
Sub Gross

   strGroesse = " einen großen Becher "

End Sub


Sub Mittel
```

```
   strGroesse = " einen mittleren Becher "
End Sub

Sub Klein
   strGroesse = " einen kleinen Becher "
End Sub
```

Das Ergebnis wird ausgelesen:

```
      MsgBox "Hallo " & ctlTxt.getText() & Chr(13) &  _
         "Nun gibt es" & strGroesse & strGetraenk
```

Die Variable wird öffentlich deklariert und muss beim Initialisieren des Dialogs gefüllt werden. Denn: Wird keiner der Radiobuttons aktiviert, so wäre die Variable leer:

```
   Dlg = CreateUnoDialog(DialogLibraries.Standard.Kaffeeautomat)
   strGroesse = " einen kleinen Becher "
   Dlg.Execute()
```

2.2.4 Markierfelder (Kontrollkästchen)

Anschließend werden zwei Kontrollkästchen auf dem Dialog erzeugt: `chkZucker` und `chkMilch`. Damit es dem Anwender nicht möglich ist, Cola und Milch beziehungsweise Brühe und Zucker auszuwählen, wird die Eigenschaft `Enable` der Kontrollkästchen auf `False` gesetzt. Dies kann in einer Prozedur ausgelagert werden, die von unterschiedlichen Optionsfeldern aufgerufen wird:

```
Sub EinAus(blnSchalter As Boolean)
   Dlg.getControl("chkZucker").Enable = blnSchalter
   Dlg.getControl("chkMilch").Enable = blnSchalter
End Sub

Sub Ein
   call EinAus(True)
End Sub

Sub Aus
   call EinAus(False)
```

```
End Sub
```

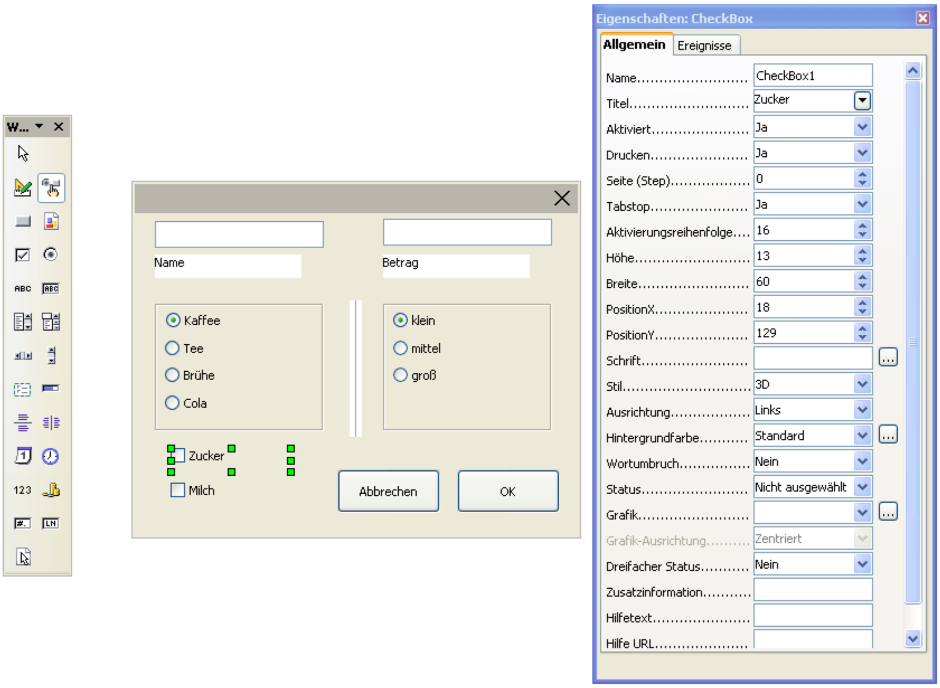


Abbildung 2.51 Kontrollkästchen

Wenn der Benutzer beim Kaffee Zucker und Milch eingeschaltet hat, er allerdings nun Cola wählt, so bleiben Zucker und Milch immer noch eingeschaltet – er hat sogar keine Möglichkeit mehr, dies auszuschalten. Deshalb muss die erste Prozedur ein wenig modifiziert werden:

```
Sub EinAus(blnSchalter As Boolean)
   Dlg.getControl("chkZucker").Enable = blnSchalter
   Dlg.getControl("chkMilch").Enable = blnSchalter
   If blnSchalter = False Then
      Dlg.getControl("chkMilch").State = False
      Dlg.getControl("chkZucker").State = False
   End If
```

```
End Sub
```

Das kann anschließend überprüft werden, beispielsweise so:

```
If Dlg.getControl("chkZucker").State = 1 _
   And Dlg.getControl("chkMilch").State = 1 Then
   strGetraenk = strGetraenk & " mit Zucker und Milch."
ElseIf Dlg.getControl("chkZucker").State = 1 Then
   strGetraenk = strGetraenk & " mit Zucker."
ElseIf Dlg.getControl("chkMilch").State = 1 Then
   strGetraenk = strGetraenk & " mit Milch."
End If
```

Beachten Sie, dass der State des Kontrollkästchens mit dem Wert 1 abgefragt wird, während Optionsfelder mir True oder False verglichen werden.

Bei der Eigenschaft „Enable“: Die Kontrollkästchen besitzen die Eigenschaft `Enable`, aber `Model.Enabled`, die Methode wiederum heißt `setEnable()`!

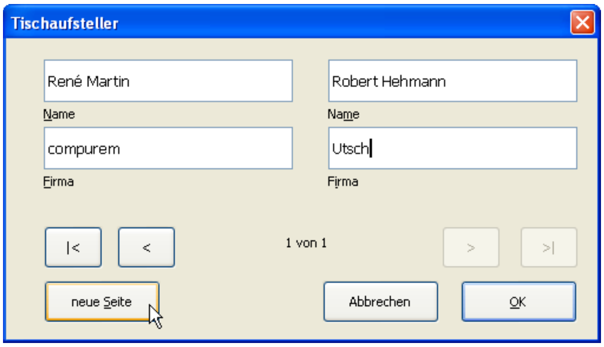
Tabelle 2.26 Einige Eigenschaften der Optionsfelder und Kontrollkästchen

Eigenschaft	Wert	Beschreibung
Label	String	Beschriftung des Optionsfeldes
Model.Height, Model.Width, Model.PositionX und Model.PositionY	Long	Höhe, Breite, X- und Y-Position des Steuerelementes (in ma-Einheiten)
Enable Model.Enabled	Boolean	das Steuerelement ist aktivierbar
Visible	Boolean	sichtbar
Model.Tabstop	Boolean	ermöglicht die Erreichbarkeit des Steuerelementes über die Tabulatortaste
Model.TabIndex	Long	Position des Steuerelementes in der Aktivierungsreihenfolge
ImplementationName	String	stardiv.Toolkit.UnoRadioButtonControl (Optionsfeld) und stardiv.Toolkit.UnoImageControlControl (grafisches Kontrollfeld)

Tabelle 2.27 Einige Methoden der Optionsfelder und Kontrollkästchen

Methode	**Wert**	**Beschreibung**
setLabel(String)	Void	ändert die Beschriftung bei Optionsfeldern
setVisible(Boolean)	Void	macht sichtbar oder unsichtbar
setEnable(Boolean)	Void	aktiviert oder schaltet die Aktivierbarkeit aus
setFocus()	Void	setzt den Fokus (Cursor)

Ein weiteres Beispiel zu Schaltflächen und Textfeldern finden Sie bei den Dateien unter „Tischaufsteller“.



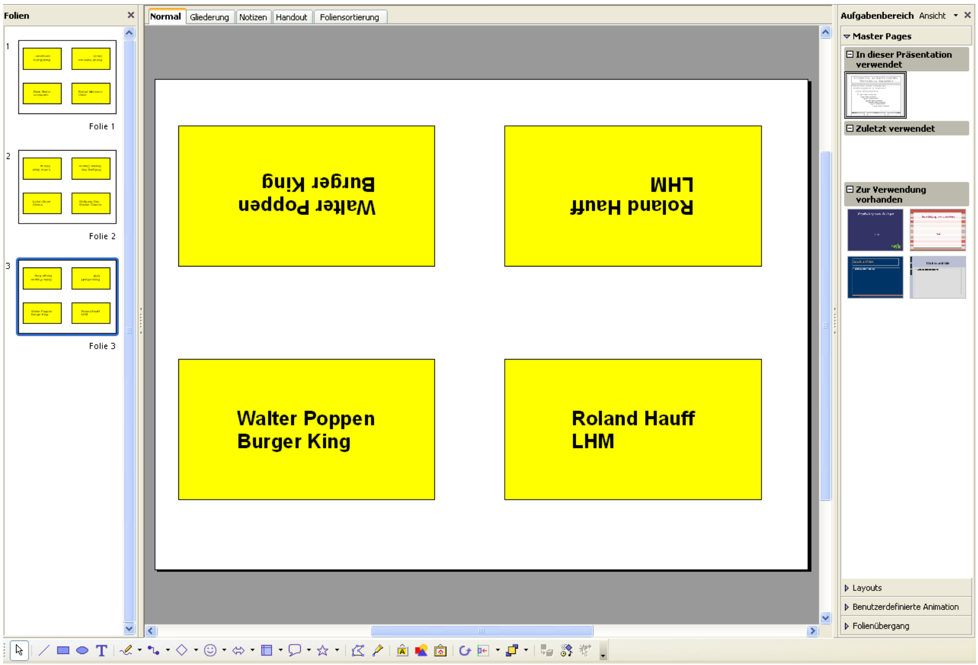


Abbildung 2.52 Über den Dialog werden die Namen für die Tischaufsteller eingegeben.

2.3 Eine Süßwarenautomat

2.3.1 Listenfelder

Auf einem zweiten Dialog werden mehrere Schaltflächen und eine Listbox angebracht. Die Listbox erhält den Namen „lstListe“. Beim Initialisieren des Dialogs wird sie gefüllt:

```
Sub SuesswarenStart


   DialogLibraries.LoadLibrary("Standard")

   Dlg = CreateUnoDialog(DialogLibraries.Standard.Suesswaren)

   Dlg.Title = "Süßwarenmaschine"

   Dlg.getControl("lstListe").addItem("Hanuta", 0)

   Dlg.getControl("lstListe").addItem("Duplo", 1)
```

```
    Dlg.getControl("lstListe").addItem("Mars", 2)
    Dlg.getControl("lstListe").addItem("Bounty", 3)
    Dlg.getControl("lstListe").addItem("Haribo", 4)
    Dlg.Execute()

End Sub
```

Der erste Parameter der Methode `addItem` gibt die Zeichenkette an, mit der die Liste gefüllt wird, der zweite die Position. Selbstverständlich würde auch ein Füllen in der umgekehrten Reihenfolge funktionieren:

```
    Dlg.getControl("lstListe").addItem("Hanuta", 0)
    Dlg.getControl("lstListe").addItem("Duplo", 0)
    Dlg.getControl("lstListe").addItem("Mars", 0)
...
```

Selbstverständlich brauchen Sie bei festen Listeneinträgen diese nicht per Programmierung füllen zu lassen, sondern können sie „hart“ in der Eigenschaft „Listeneinträge“ eingeben. Das hier beschriebenen Verfahren macht nur dann Sinn, wenn die Werte beispielsweise aus einer Datenbank oder Tabelle, die sich ändern kann, ausgelesen werden sollen.

Die Eigenschaft `ItemCount` liefert die Anzahl der Einträge. Sie kann ermittelt werden:

```
    MsgBox Dlg.getControl("lstListe").ItemCount
```

Über eine Schaltfläche kann ein neuer Eintrag hinzugefügt werden:

```
Sub cmd_Neu
    Dim intAnzahl As Integer
    Dim lstListe As Object

    lstListe = Dlg.getControl("lstListe")
    intAnzahl = lstListe.ItemCount
    lstListe.addItem("Überraschungsei" & intAnzahl + 1, intAnzahl)

End Sub
```

Die Eigenschaft `selectedItem` liefert den Text des markierten Eintrags:

```
Sub cmd_WelcherAusgewaehlt

    Dim lstListe As Object
```

```
   lstListe = Dlg.getControl("lstListe")
   MsgBox lstListe.selectedItem

End Sub
```

Die Eigenschaft `selectedItemPos` liefert die Nummer des markierten Eintrags:

```
   MsgBox lstListe.selectedItem, , _
      "ausgewählte Nummer: " & lstListe.selectedItemPos
```

Die Zählung beginnt bei 0. Wurde nichts markiert, dann ergibt `selectedItem` einen leeren String, `selectedItemPos` die Zahl -1. Somit kann überprüft werden, ob der Anwender noch nichts markiert hat.

Die Methoden `selectItem` und `selectItemPos` setzen oder löschen eine Markierung. Der zweite, boolesche Parameter stellt dabei den Schalter dar, ob der Eintrag gesetzt oder gelöscht wird:

```
   lstListe.selectItem("Hanuta", True)
   lstListe.selectItemPos(0, True)
```

Existiert der Eintrag nicht, dann wird – ohne Fehlermeldung – nichts markiert.

Soll der markierte Eintrag gelöscht werden, dann stellt die Listbox die Methode `removeItems` zur Verfügung: Sie löscht ab einer bestimmten Position eine Anzahl von Einträgen:

```
Sub cmd_EinerWeg

   Dim lstListe As Object
   Dim intPos As Integer

   lstListe = Dlg.getControl("lstListe")
   intPos = lstListe.selectedItemPos
   If intPos > -1 Then
      lstListe.removeItems(intPos, 1)
   Else
      MsgBox "Bitte markieren Sie einen Eintrag!"
   End If

End Sub
```

Und damit wird klar, wie alle Einträge gelöscht werden können:

```
Sub cmd_AlleWeg

   Dim lstListe As Object
   Dim intAnzahl As Integer

   lstListe = Dlg.getControl("lstListe")
   intAnzahl = lstListe.ItemCount
   lstListe.removeItems(0, intAnzahl)

End Sub
```

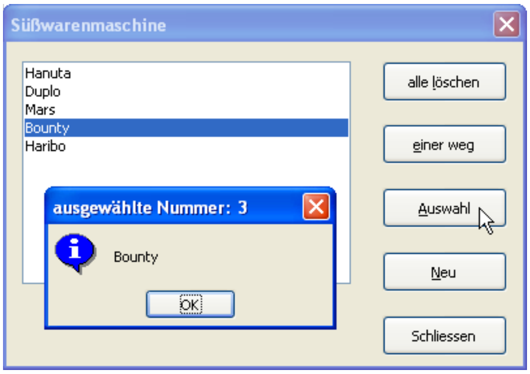


Abbildung 2.53 Das Listenfeld

2.3.2 Kombinationsfelder

Am unteren Rand befindet sich ein Kombinationsfeld (Combobox, Dropdown-Box). Es wird mit Einträgen gefüllt. Damit der erste Eintrag nach der Anzeige sichtbar ist, wird er mit der Eigenschaft Text „gefüllt“:

```
Dlg.getControl("cboArt").addItem("Einfachauswahl", 0)
Dlg.getControl("cboArt").addItem("Mehrfachauswahl", 1)
Dlg.getControl("cboArt").Text = "Einfachauswahl"
```

Da die Methode `getItem` den Text zurückgibt, der sich hinter einer Nummer verbirgt, könnte es auch mit

```
   Dlg.getControl("cboArt").Text = Dlg.getControl("cboArt").getItem(0)
```

gelöst werden.

Wird ein anderer Eintrag ausgewählt, dann soll sich die Eigenschaft der Listbox ändern:

```
Sub cbo_Wechsel

   Dim cboArt As Object

   cboArt = Dlg.getControl("cboArt")
   If cboArt.SelectedText = "Einfachauswahl" Then
      Dlg.getControl("lstListe").MultipleMode = False
   ElseIf cboArt.SelectedText = "Mehrfachauswahl" Then
      Dlg.getControl("lstListe").MultipleMode = True
   End If
End Sub
```

Dieses Ereignis wird beim „Auslösen“ der Combobox aktiviert. Nun kann der Benutzer entweder nur einen oder mehrere Einträge auswählen. Die beiden Schaltflächen „Neu“ und „alle löschen“ funktionieren noch immer, jedoch „Auswahl“ und „einer weg“ zeigen beziehungsweise löschen jeweils nur den zuerst markierten.

Bei der Auswahl-Schaltfläche wird nun differenziert – wurde Einzel- oder Mehrfachmodus eingeschaltet? Während die Methode `MultipleMode` diese Option einschaltet, kann sie mit der Eigenschaft `isMutipleMode` abgefragt werden.

Hier liegt mit Sicherheit ein Schreibfehler vor – diese Eigenschaft heißt „isMutipleMode“ und nicht „isMultipleMode“!

Die Eigenschaft `SelectedItems` und die Methode `getSelectedItems()` übergeben sämtliche Einträge an ein Datenfeld, das anschließend ausgelesen werden kann:

```
Sub cmd_WelcherAusgewaehlt

   Dim lstListe As Object
   Dim strEintraege() As String
   Dim strListe As String
   Dim intPos As Integer
```

```
    lstListe = Dlg.getControl("lstListe")
    If lstListe.isMutipleMode  = False Then
       MsgBox lstListe.selectedItem, , _
          "ausgewählte Nummer: " & lstListe.selectedItemPos
    Else
       strEintraege = lstLIste.SelectedItems
       For intPos = LBound(strEintraege) To UBound(strEintraege)
          strListe = strListe & Chr(13) & strEintraege(intPos)
       Next
       MsgBox strListe
    End If

End Sub
```

Analog können auch die Nummern mit der Methode `getSelectedItemsPos()` an eine Variable übergeben werden.

Mit diesem Wissen ist es nicht mehr schwierig, das Löschen mehrerer selektierter Einträge durchzuführen. Beachten Sie jedoch, dass die Zählung vom letzten zum ersten vorgenommen werden muss!

```
Sub cmd_EinerWeg
    Dim lstListe As Object
    Dim intEintraege() As Integer
    Dim intPos As Integer

    lstListe = Dlg.getControl("lstListe")
    If lstListe.isMutipleMode = False Then
       intPos = lstListe.selectedItemPos
       If intPos > -1 Then
          lstListe.removeItems(intPos, 1)
       Else
          MsgBox "Bitte markieren Sie einen Eintrag!"
       End If
    Else
```

```
      intEintraege = lstLIste.getSelectedItemsPos()
      For intPos = UBound(intEintraege) To LBound(intEintraege) Step -1
         lstListe.removeItems(intEintraege(intPos), 1)
      Next
   End If
End Sub
```

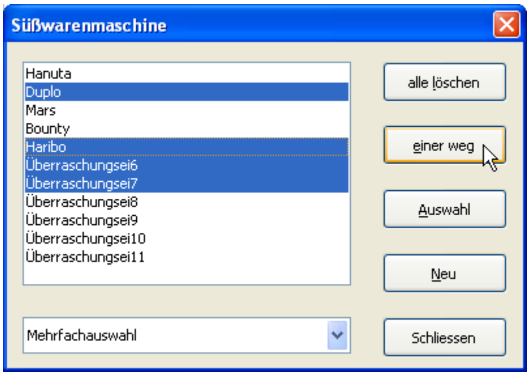


Abbildung 2.54 Mehrere Einträge können gleichzeitig gewählt und gelöscht werden.

Tabelle 2.28 Wichtige Eigenschaften der Listbox

Eigenschaft	**Wert**	**Beschreibung**
ItemCount	Integer	Die Anzahl der Elemente (nur lesbar)
Items	Array	Sämtliche Elemente (nur lesbar)
SelectedItemPos	Integer	Die Nummer des markierten Eintrags (nur lesbar)
SelectedItemsPos	Array	Die Nummern sämtlicher markierter Einträge (nur lesbar)
SelectedItem	String	Text des markierten Eintrags (nur lesbar)
SelectedItems	Array	Die Texte sämtlicher markierter Einträge (nur lesbar)
MultipleMode	Boolean	Mehrfachauswahl: True aktiviert die Möglichkeit zur Mehrfachauswahl von Einträgen, False sperrt die Mehrfachauswahl (die Eigenschaft kann nicht direkt, sondern nur über IsMutipleMode ausgelesen werden).
Visible	Boolean	Sichtbar
Enable Model.Enabled	Boolean	Aktivierbar

Tabelle 2.29 Wichtige Methoden der Listbox

Methode	Wert	Beschreibung
addItem(String, Integer)	Void	fügt die angegebene Zeichenfolge an der angegebenen Position in die Liste ein
addItems(Array, Integer)	Void	fügt die in dem Zeichenfolgen-Datenfeld ItemArray aufgeführten Einträge in die Liste an der angegebenen Position ein
removeItems(Integer, Integer)	Void	entfernt eine Anzahl Einträge ab der angegebenen Position
getItemCount()	Integer	ermittelt die Anzahl der Einträge
getItem(Integer)	String	ermittelt den Eintrag zu einer bestimmten Nummer
getSelectedItemPos()	Integer	ermittelt die Nummer des markierten Eintrags
getSelectedItemsPos()	Array	ermittelt die Nummern der markierten Einträge
getSelectedItem()	String	ermittelt den Text des markierten Eintrags
getSelectedItems()	Array	ermittelt die Texte der markierten Einträge
selectItemPos(Integer, Boolean)	Void	schaltet die Markierung für das angegebene Element (Nummer) ein oder aus
selectItemsPos(Array, Boolean)	Void	schaltet die Markierung für die angegebenen Elemente (Nummern) ein oder aus
selectItem(String, Boolean)	Void	schaltet die Markierung für das angegebene Element (Text) ein oder aus
isMutipleMode()	Boolean	erlaubt eine Mehrfachauswahl innerhalb der Liste (nur lesbar)
setMultipleMode(Boolean)	Void	schaltet die Mehrfachauswahl ein oder aus
makeVisible(Integer)	Void	scrollt das Listenfeld so, dass der über Pos angegebene Eintrag sichtbar wird
setFocus()	Void	setzt den Fokus

Tabelle 2.30 Wichtige Eigenschaften der Combobox

Eigenschaft	Wert	Beschreibung
ItemCount	Integer	Die Anzahl der Elemente (nur lesbar)
Items	Array	Sämtliche Elemente (nur lesbar)

Eigenschaft	Wert	Beschreibung
DropDownLineCount	Integer	Die Anzahl der Elemente, wenn die Liste aufgeklappt wird
Text	String	Der angezeigte Text
SelectedText	String	Der ausgewählte Text (nur lesbar)
Visible	Boolean	Sichtbar
Enable Model.Enabled	Boolean	Aktivierbar
Editable	Boolean	Änderbar

Tabelle 2.31 Wichtige Methoden der Combobox

Methode	Wert	Beschreibung
addItem(String, Integer)	Void	fügt die angegebene Zeichenfolge an der angegebenen Position in die Liste ein
addItems(Array, Integer)	Void	fügt die in dem Zeichenfolgen-Datenfeld ItemArray aufgeführten Einträge in die Liste an der angegebenen Position ein
removeItems(Integer, Integer)	Void	entfernt eine Anzahl Einträge ab der angegebenen Position
getItemCount()	Integer	ermittelt die Anzahl der Einträge
getDropDownLineCount()	Integer	die Anzahl der Elemente, wenn die Liste aufgeklappt wird
setDropDownLineCount(Integer)	Void	die Anzahl der Elemente, wenn die Liste aufgeklappt wird
getSelectedText()	String	der markierte Text
isEditable()	Boolean	änderbar
setEditable(Boolean)	Void	Liste kann verändert werden
setVisible(Boolean)	Void	Liste wird sichtbar
setEnable(Boolean)	Void	Liste kann geändert werden
setFocus()	Void	erhält den Fokus

In vielen Microsoft-Produkten besteht der Unterschied zwischen Listenfeldern und Kombinationsfeldern darin, dass Listenfelder eine Mehrfachauswahl zulassen können, während Kombinationsfelder das Eintragen von eigenen Werten, die nicht in der Liste stehen, ermöglichen. Anders jedoch in OOo-Basic: Kombinationsfelder können immer mit benutzerdefinierten Einträgen gefüllt werden. Die Eigenschaft „Max. Textlänge" kann ledig-

lich verhindern, dass der eingegebene Text eine bestimmte Zeichenzahl überschreitet. Soll dies verhindert werden, muss ein Listenfeld verwendet werden, die Mehrfachselektion auf „nein“ und „Aufklappbar“ auf „ja“ gesetzt werden. Dann sollte es natürlich von seiner „offenen“ Form auf eine „geschlossene“ Form zusammengeschoben werden, denn nur dann funktioniert das Aufklappen. Und: Selbst wenn Sie nun die Eigenschaft „Mehrfachselektion“ auf „ja“ setzen, ist es nicht möglich, in der Listenbox, die aussieht wie ein Kombinationsfeld, mehrere Einträge auszuwählen.

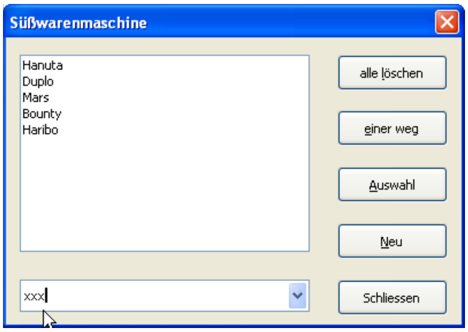


Abbildung 2.55 Der Eintrag in der Combobox könnte überschrieben werden.

2.4 Ein alberner Scherz (?)

Auf einem Dialog befindet sich eine Befehlsschaltfläche. Der Dialog ist beschriftet mit „Wenn Sie hier klicken, dann können Sie viel Geld gewinnen“. Versucht nun der Benutzer, auf die Schaltfläche zu klicken, dann hüpft sie nach rechts. Versucht er dort, sie anzuklicken, dann springt sie nach links. Für dieses „Problem“ existieren mehrere unterschiedliche Lösungen.

Die erste Lösung arbeitet mit zwei Befehlsschaltflächen. Eine von beiden ist unsichtbar. Fährt der Mauszeiger über die sichtbare (das auslösende Ereignis ist „Mausbewegung“), dann wird die zweite sichtbar, während die erste den Wert `Visible` auf „False“ setzt. Bei der zweiten Schaltfläche wird das Verfahren umgekehrt angewandt:

```
Sub cmdOK1_MouseOver

   Dlg.getControl("cmdOK2").Visible = True

   Dlg.getControl("cmdOK1").Visible = False

End Sub
```

```
Sub cmdOK2_MouseOver
   Dlg.getControl("cmdOK2").Visible = False
   Dlg.getControl("cmdOK1").Visible = True
End Sub
```

Die zweite Variante lässt den Button echt springen. Sitzt er links, dann wird er nach rechts verschoben, falls nicht, so hüpft er wieder zurück. Das auslösende Ereignis ist erneut „Mausbewegung“:

```
Sub cmdOK3_MouseOver
   Dim ctl As Object

   ctl = Dlg.getControl("cmdOK3")

   If ctl.PosSize.X = 20 Then
      ctl.SetPosSize(260, 83, 250, 83, 1)
   Else
      ctl.SetPosSize(20, 83, 250, 83, 1)
   End If

End Sub
```

Es geht auch folgendermaßen:

```
Sub cmdOK4_MouseOver
   Dim ctl As Object
   ctl = Dlg.getControl("cmdOK4")
   If ctl.Model.PositionX = 10 Then
      ctl.Model.PositionX = 149
   Else
      ctl.Model.PositionX = 10
   End If

End Sub
```

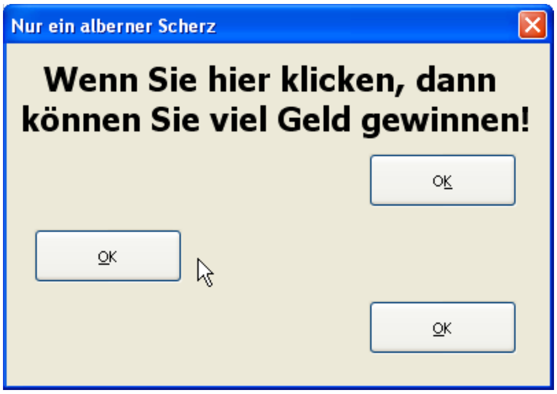


Abbildung 2.56 Hüpfende Buttons …

2.5 Mehrseitige Dialoge

Eine interessante, wenn auch etwas mühevoll zu erstellende Technik stellt OOo über die mehrseitigen Dialoge zur Verfügung. Alle Steuerelemente, die auf der Seite („Step“) mit der Nummer 0 liegen, werden immer angezeigt. Umgekehrt werden auf der Seite 0 alle Steuerelemente – unabhängig von ihrer Nummer – angezeigt. Liegen sie jedoch auf einer anderen Seite, so werden sie nur beim Aufruf dieser Seite sichtbar, auf der nächsten und vorhergehenden verschwinden sie wieder. Diese Eigenschaft kann abgefragt und gesetzt werden über:

```
Dlg.Model.Step
```

Ein wenig problematisch ist diese Technik, da zu Beginn die Seite 0 angezeigt wird, also alle Steuerelemente. In den nächsten Schritten werden jeweils einige ausgeblendet. Deshalb empfiehlt es sich, wenn Sie mit den Seiten arbeiten, bei der Seite 1 zu beginnen.

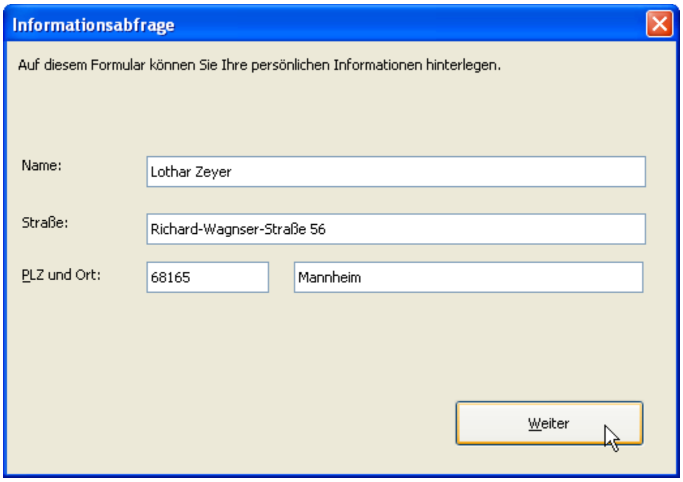


Abbildung 2.57 Die Startseite

Im unten stehenden Dialog liegt das Label auf der Seite 0, ist also immer sichtbar. Die übrigen Labels, Textfelder und das Listenfeld befinden sich auf den Seiten 1, 2 oder 3. Beim Start wird die Liste gefüllt:

```
Sub Main

   DialogLibraries.LoadLibrary("Standard")
   Dlg = CreateUnoDialog(DialogLibraries.Standard.dlgMehrereSeiten)
   lblInfo = Dlg.getControl("lblInfo")

    Dlg.getControl("lstListe").addItem("Writer", 0)
    Dlg.getControl("lstListe").addItem("Calc", 1)
    Dlg.getControl("lstListe").addItem("Draw", 2)
    Dlg.getControl("lstListe").addItem("Base", 3)
    Dlg.getControl("lstListe").addItem("Impress", 4)

    Dlg.getControl("txtName").setFocus()

   Dlg.Model.Step = 1
   Call Anzeige
```

```
    Dlg.Execute()

End Sub
```

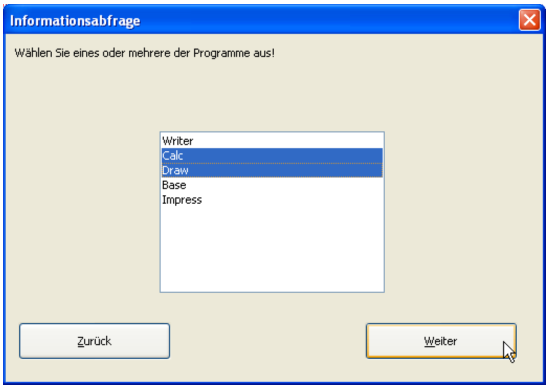


Abbildung 2.58 Die zweite Seite

Die Prozedur „Anzeige“ überprüft, auf welcher Seite sich die Anzeige befindet, ändert die Beschriftung der Befehlsschaltfläche „Weiter“ in „Fertig stellen“, den Text des überall sichtbaren Labels, und blendet auf der ersten Seite den Zurück-Button aus. Es ist nicht nötig, ihn explizit einzublenden, da dies über die Seiten geschieht.

```
Sub Anzeige

    Select Case Dlg.Model.Step
    Case 1
        lblInfo.Text = "Auf diesem Formular können Sie Ihre " & _
            "persönlichen Informationen hinterlegen."
        Dlg.getControl("cmdZurueck").Visible = False
        Dlg.getControl("cmdWeiter").Label = "Weiter"
    Case 2
        lblInfo.Text = "Wählen Sie eines oder mehrere der Programme aus!"
        Dlg.getControl("cmdWeiter").Label = "Weiter"
    Case 3
        lblInfo.Text = "Bitte geben Sie einen persönlichen Kommentar ein!"
        Dlg.getControl("cmdWeiter").Label = "fertig stellen"
```

```
   End Select

End Sub
```

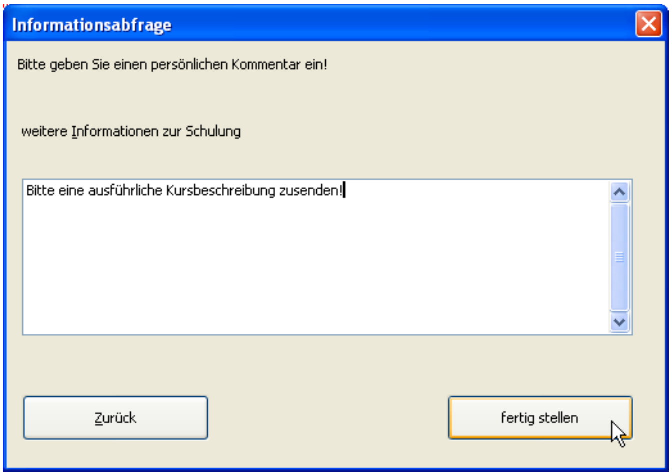


Abbildung 2.59 Die dritte Seite

Die Zurück-Schaltfläche verkleinert die Nummer der Seite um 1:

```
Sub cmdZurueck_Click

   If Dlg.Model.Step > 1 Then
      Dlg.Model.Step = Dlg.Model.Step - 1
   End If
   Call Anzeige

End Sub
```

Die Weiter-Schaltfläche springt zum nächsten Schritt:

```
Sub cmdWeiter_Click
   Dim strEintraege() As String
   Dim strMeldung As String
   Dim intPos As Integer
```

```
  If Dlg.Model.Step < 3 Then
    Dlg.Model.Step = Dlg.Model.Step + 1
  Else
```

Wurden alle Informationen eingegeben, dann kann der Assistent die Daten auswerten und weiter verarbeiten:

```
  Else
    strMeldung = "Name:    " & Dlg.getControl("txtName").getText() _
      & Chr(13)
    strMeldung = strMeldung & "Straße:  " & _
      Dlg.getControl("txtStrasse").getText() & Chr(13)
    strMeldung = strMeldung & "Plz/Ort: " & _
      Dlg.getControl("txtPlz").getText() & " " & _
      Dlg.getControl("txtOrt").getText() & Chr(13) & Chr(13)
    strMeldung = strMeldung & "Programm(e):" & Chr(13)

    lstListe = Dlg.getControl("lstListe")
    strEintraege = lstLIste.SelectedItems
    For intPos = LBound(strEintraege) To UBound(strEintraege)
      strMeldung = strMeldung & Chr(13) & "* " & strEintraege(intPos)
    Next

    strMeldung = strMeldung & Chr(13) & Chr(13) & _
      Dlg.getControl("txtWeitereInfos").getText()

    MsgBox strMeldung
    Dlg.endExecute()
  End If

  Call Anzeige

End Sub
```

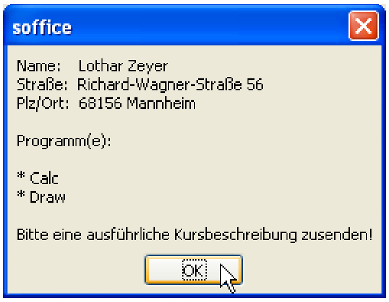


Abbildung 2.60 Die kompletten Informationen

2.5.1 Grafische Kontrollfelder (Imagecontrols)

Mit Hilfe der Seiten kann übrigens leicht eine Diashow erzeugt werden:

```
   Public DlgBilder As Object

   Public intObergrenze As Integer



Sub Main

   intObergrenze = 10

   DialogLibraries.LoadLibrary("Standard")

   DlgBilder = CreateUnoDialog(DialogLibraries.Standard.Diashow)

   DlgBilder.Model.Step = 1


   DlgBilder.Execute()


End Sub


Sub cmdDiaWeiter_Click

   If DlgBilder.Model.Step  < intObergrenze Then

      DlgBilder.Model.Step = DlgBilder.Model.Step +1

   Else

      DlgBilder.Model.Step = 1
```

```
   End If

End Sub
```

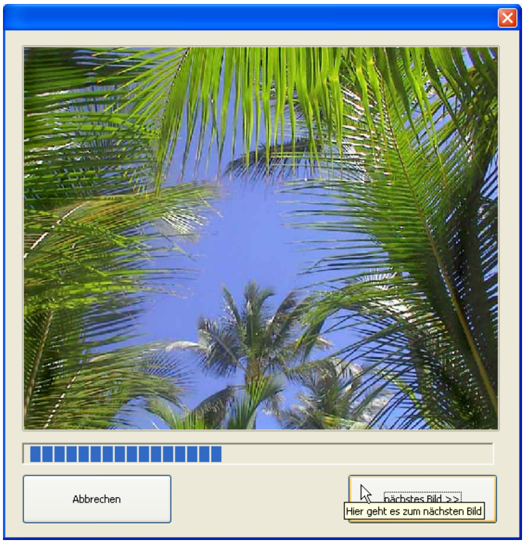


Abbildung 2.61 Die Diashow

2.5.2 Fortschrittsbalken (Progessbar)

Ähnlich, wie es über das grafische Kontrollfeld wenig zu sagen gibt, bleibt auch für die übrigen Kontrollfelder nicht viel zu sagen. Beim Fortschrittsbalken werden die Eigenschaften Fortschrittswert, minimaler und maximaler Fortschrittswert über den Eigenschaftendialog oder per Programmierung festgelegt:

Sie können gesetzt werden:

```
Dlg.getControl("ProgressBar1").Model.ProgressValueMin = 0

Dlg.getControl("ProgressBar1").Model.ProgressValueMax = 1000
```

Der aktuelle Wert wird abgefragt oder gesetzt über die Eigenschaft:

```
Dlg.getControl("ProgressBar1").Value
```

oder über die Methoden:

```
Dlg.getControl("ProgressBar1").setValue(Long)
```

```
Dlg.getControl("ProgressBar1").getValue()
```

Folglich kann der Fortschrittsbalken folgendermaßen „weitergezählt“ werden:

```
DlgBilder.getControl("ProgressBar1").Value = _
   DlgBilder.getControl("ProgressBar1").Value + 1
```

Wird der maximale Fortschrittswert überschritten, dann erfolgt keine Fehlermeldung, sondern der Balken bleibt am äußersten Rand stehen. Dies ist eine bemerkenswerte Eigenschaft! Sie sollten also darauf achten, dass Anfang und Ende synchron mit dem Ablauf Ihres Programms zusammenfällt.

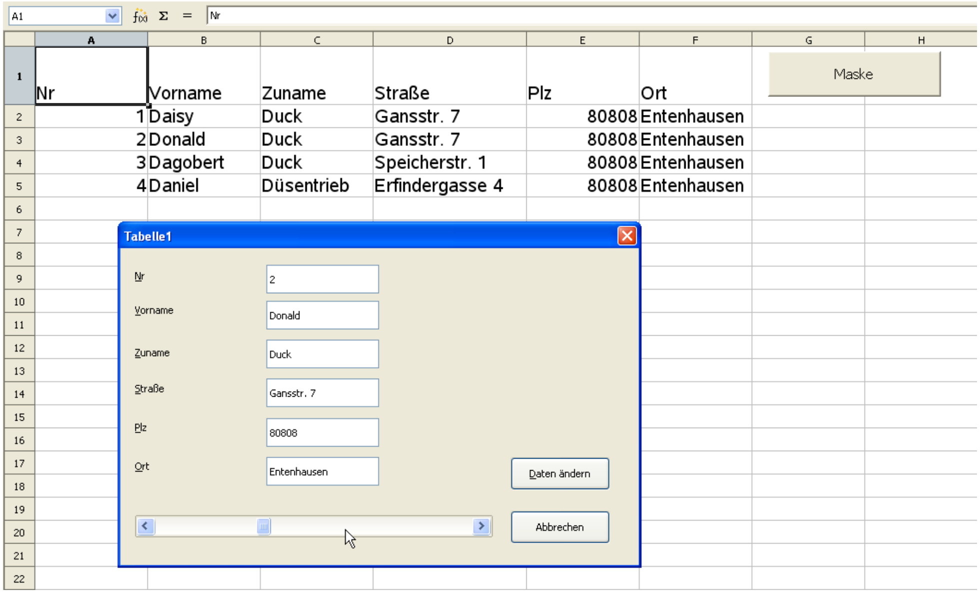


Abbildung 2.62 Ein Fortschrittsbalken bei einer datenbankähnlichen Anwendung

2.5.3 Währungsfelder, numerische und formatierte Felder

Das Verwenden eines Textfeldes im Beispiel des Kaffeeautomaten war überflüssig. Geschickter wäre ein numerisches Feld gewesen. Dort können minimaler und maximaler Wert eingestellt werden, die Anzahl der Nachkommastellen und Tausendertrennzeichen.

Beachten Sie, dass der Inhalt des numerischen Feldes mit der Eigenschaft `Value` und nicht mit `Text` abgefragt wird.

Wenn der Benutzer eine Zahl mit Nachkommastellen eingibt, dann wird sie zuerst im Sinne der Eigenschaft „Nachkommastellen“ gerundet und anschließend das (kaufmännisch) gerundete Ergebnis angezeigt!

Die numerischen Felder stellen eine wertvolle Hilfe dar, wenn in einem Textfeld Zahlen eingegeben werden sollen. Alles, was keine Zahl ist, wie beispielsweise:

- 1ooo
- l000
- 12,50,50
- 100 Euro
- 25,--

und so weiter, wird unterbunden. Es ist dem Anwender auch nicht möglich, per Copy & Paste einen Text in das Zahlenfeld einzufügen – jegliche „falsche“ Eingabe wird verweigert.

Übrigens: Wird ein Drehfeld eingeschaltet, dann hat der Anwender die Möglichkeit, Zahlen „hochzudrehen“.

Hat der Benutzer nichts eingegeben, dann wird der Wert 0 ausgegeben.

Währungsfelder unterscheiden sich von ihren numerischen Kollegen lediglich dadurch, dass im Feld ein (vorgegebenes) Währungszeichen angezeigt wird. Wird der Inhalt abgefragt, dann wird nur die Zahl ausgelesen.

Noch komplexere Formatierungen können über formatierte Felder eingestellt werden. Mit der Eigenschaft Text kann der angezeigte Text ausgelesen werden, mit `EffectiveValue` wird der eigentliche Wert ausgelesen.

2.5.4 Datumsfelder und Uhrzeitfelder

Das Datumsfeld verfügt neben einem Drehfeld, mit dem das Datum um einen Tag weiter- oder zurückgedreht werden kann, auch über die Eigenschaft „aufklappbar“, mit dem ein Kalender angezeigt werden kann. Der Inhalt des Datumsfeldes wird mit der Methode `getDate()` oder der Eigenschaft `Model.Date` ausgelesen.

Ein Datum wie beispielsweise der 24.12.2005 wird als 20051224 zurückgegeben. Damit dieses Datum als Datum weiter verarbeitet werden kann, muss es umgewandelt werden. Dies geht über die Funktion `CDateFromIso`. Sie liefert nun korrekt den 24.12.2005. Wird umgekehrt ein Datum in das Datumsfeld eines Dialogs zurückgeschrieben, dann

muss es mit `CDateToIso` umgewandelt werden. Ein Beispiel für ein ausgewertetes Datum finden Sie in Kapitel 9.

Aus einem Zeitfeld, für das es nur ein Drehfeld, aber kein Vorgabefeld zum Aufklappen gibt, kann der Wert über die Methode `getTime()` oder die Eigenschaft `Model.Time` ausgelesen werden. Eine als 14:25 eingegebene Uhrzeit wird als 14250000 zurückgegeben. Leider existiert keine Umwandlungsfunktion, damit diese Zeit „vernünftig“ weiter verarbeitet werden kann. Man müsste eine Prozedur schreiben:

```
Sub ZeitAuslesen
   Dim strZeit As String

   strZeit = Dlg.getControl("TimeField1").getTime()
   strZeit = Left(strzeit, Len(strZeit) - 4)
   strZeit = "Es wurde eingegeben: " & _
      Left(strZeit, Len(strZeit) - 2) & " Stunde(n) und " & _
      Right(strZeit, 2) & " Minute(n)."
   MsgBox strZeit
End Sub
```

Die Zahl 4 im Code ergibt sich hierbei aus dem Zeitformat „13:45“.

2.5.5 Dateiauswahl

Auch zum Kontrollfeld „Dateiauswahl“ bleibt nicht viel zu sagen: Klickt der Benutzer auf das Feld mit den drei Pünktchen, dann öffnet sich der Öffnen-Dialog. Wird nun eine Datei ausgewählt, dann steht der vollständige Name im Feld (auch wenn er möglicherweise nicht sichtbar ist).

Um sicherzugehen, dass der Internet-Standard RFC 1738 eingehalten wird, sollte die Funktion `ConvertToUrl` verwendet werden.

Leider existiert kein Kontrollfeld, mit dem sich ein Ordner (oder Laufwerk) auswählen lässt. Wie man dies lösen kann, wird in Kapitel 9 beschrieben.

2.5.6 Maskiertes Feld

Interessante Möglichkeiten lässt das maskierte Feld zu. Wird dort die Eigenschaft „Formatüberprüfung“ auf „ja“ gesetzt, dann kann im Feld „Eingabemaske“ und „Zeichenmaske“ eine Zeichenfolge eingegeben werden. Für die Eingabemaske stehen folgende Zeichen zur Verfügung:

Tabelle 2.32 Zeichen der Eingabemaske des maskierten Feldes

Zeichen	Bedeutung
L	Konstante, die nicht geändert werden kann
a	alle Buchstaben a - z und A - Z sind zugelassen
A	alle Buchstaben a - z und A - Z sind zugelassen; Kleinbuchstaben werden jedoch in Großbuchstaben verwandelt
c	alle Buchstaben a - z und A - Z und Ziffern 0 - 9 sind zugelassen
C	alle Buchstaben a - z und A - Z und Ziffern 0 - 9 sind zugelassen; Kleinbuchstaben werden jedoch in Großbuchstaben verwandelt
N	alle Ziffern 0 - 9 sind zugelassen
x	alle Zeichen sind zugelassen
X	alle Zeichen sind zugelassen, Kleinbuchstaben werden jedoch in Großbuchstaben verwandelt

Steht beispielsweise in „Eingabemaske“ die Zeichenfolge „NN-NN-NNNN“ und in „Formatmaske“ die Zeichenmaske „__-__-____“, dann kann der Benutzer auf den vorgegebenen Unterstrichen eine beliebige Ziffernkombination eingeben.

2.6 Auf viele Kontrollelemente zugreifen

Wenn Sie per Programmierung auf mehrere Kontrollelemente zugreifen möchten (beispielsweise um sie dynamisch ein- oder auszublenden oder Inhalte aus Textfelder auslesen möchten), können Sie dies mit einer Schleife organisieren:

```
Sub Unsichtbar(strService As String, blnSichtbar As Boolean)
   Dim intZaehler As Integer

   For intZaehler = 0 To UBound(Dlg.Controls)
      If Dlg.Controls(intZaehler).Implementationname = _
         "stardiv.Toolkit." & strService Then
         Dlg.Controls(intZaehler).Visible = blnSichtbar
      End If
   Next
End Sub
```

An diese Prozedur wird der Service übergeben und das Attribut, ob das Kontrollelement sichtbar oder unsichtbar ist. Diese Prozedur kann dann von mehreren Stellen aus aufgerufen werden, beispielsweise so:

```
...
   Call Unsichtbar("UnoListBoxControl", False)
   Call Unsichtbar("UnoFixedTextControl", False)
...
```

Das komplette Beispiel finden Sie in Kapitel 11.

2.7 Zusammenfassung der Models Properties of object "stardiv.Toolkit. ":

Die folgenden beiden Tabellen listen sämtliche Eigenschaften aller Kontrollelemente der Eigenschaft „Model“ auf. Diese Übersicht dient zur Hilfe und Orientierung:

	Wert	**UnoControlDialogModel**	**UnoControlButtonModel**	**UnoControlFixedTextModel (Label)**	**UnoControlEditModel (Textfeld)**	**UnoControlDateFieldModel**	**UnoControlTimeFieldModel**	**UnoControlNumericFieldModel**	**UnoControlCurrencyFieldModel**	**UnoControlFormattedFieldModel**	**UnoControlPatternFieldModel**
Align	Integer			✓	✓					✓	
AvailableServiceNames	Array	✓									
BackgroundColor	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Border	Integer			✓	✓	✓	✓	✓	✓	✓	✓
Closeable	Boolean	✓									
ControlModels	Array	✓									
CurrencySymbol	String								✓		
Date	Long					✓					
DateFormat	Integer					✓					
DateMax	Long					✓					

	Wert	UnoControlDialogModel	UnoControlButtonModel	UnoControlFixedTextModel (Label)	UnoControlEditModel (Textfeld)	UnoControlDateFieldModel	UnoControlTimeFieldModel	UnoControlNumericFieldModel	UnoControlCurrencyFieldModel	UnoControlFormattedFieldModel	UnoControlPatternFieldModel
DateMin	Long					✓					
DateShowCentury	Boolean					✓					
DecimalAccuracy	Integer							✓	✓		
DefaultControl	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DefaultButton	Boolean		✓								
Delegator	Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Dropdown	Boolean					✓					
EchoChar	Integer				✓						
EditMask	String										✓
EffectiveDefault	Variant									✓	
EffectiveMax	Double									✓	
EffectiveMin	Double									✓	
EffectiveValue	Variant									✓	
ElementNames	Array	✓									
ElementType	Object	✓									
Enabled	Boolean	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Events	Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontCharset	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontCharWidth	Single	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontDescriptor	Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontEmphasisMark	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontFamily	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontHeight	Single	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontKerning	Boolean	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

	Wert	UnoControlDialogModel	UnoControlButtonModel	UnoControlFixedTextModel (Label)	UnoControlEditModel (Textfeld)	UnoControlDateFieldModel	UnoControlTimeFieldModel	UnoControlNumericFieldModel	UnoControlCurrencyFieldModel	UnoControlFormattedFieldModel	UnoControlPatternFieldModel
FontName	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontOrientation	Single	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontPitch	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontRelief	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontSlant	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontStrikeout	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontStyleName	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontType	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontUnderline	Integer	✓	✓	✓	✓✓	✓	✓	✓	✓	✓	✓
FontWeight	Single	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontWidth	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FontWordLineMode	Boolean	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FormatKey	Long									✓	
FormatSupplier	Object									✓	
GroupControl	Boolean	✓									
GroupCount	Long	✓									
Height	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HelpText	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HelpURL	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HardLineBreaks	Boolean				✓						
HScroll	Boolean				✓						
ImageAlign	Integer		✓								
ImageURL	String		✓								
ImplementationId	Array	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

	Wert	UnoControlDialogModel	UnoControlButtonModel	UnoControlFixedTextModel (Label)	UnoControlEditModel (Textfeld)	UnoControlDateFieldModel	UnoControlTimeFieldModel	UnoControlNumericFieldModel	UnoControlCurrencyFieldModel	UnoControlFormattedFieldModel	UnoControlPatternFieldModel
ImplementationName	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Label	String		✓	✓							
LiteralMask	String										✓
MaxTextLen	Integer				✓					✓	✓
Moveable	Boolean	✓									
MultiLine	Boolean			✓	✓						
Name	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PositionX	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PositionY	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PrependCurrencySymbol	Boolean								✓		
Printable	Boolean		✓	✓	✓	✓	✓	✓	✓	✓	✓
PropertySetInfo	Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PropertyToDefault	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PushButtonType	Integer		✓								
ReadOnly	Boolean				✓		✓	✓	✓	✓	✓
ServiceName	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ShowThousends-Seperator	Boolean							✓	✓		
Sizeable	Booelan	✓									
Spin	Boolean					✓	✓	✓	✓	✓	
State	Integer		✓								
Step	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
StrictFormat	Boolean					✓	✓	✓	✓	✓	✓
SupportedServiceNames	Array	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TabIndex	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

	Wert	UnoControlDialogModel	UnoControlButtonModel	UnoControlFixedTextModel (Label)	UnoControlEditModel (Textfeld)	UnoControlDateFieldModel	UnoControlTimeFieldModel	UnoControlNumericFieldModel	UnoControlCurrencyFieldModel	UnoControlFormattedFieldModel	UnoControlPatternFieldModel
TabStop	Boolean		✓	✓	✓	✓	✓	✓	✓	✓	✓
Tag	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Text	String				✓					✓	✓
TextColor	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TextLineColor	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Time	Long						✓				
TimeFormat	Integer						✓				
TimeMax	Long						✓				
TimeMin	Long						✓				
Title	String	✓									
TreatAsNumber	Boolean									✓	
Types	Array	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VScroll	Boolean				✓						
Value	Double							✓	✓		
ValueMax	Double							✓	✓		
ValueMin	Double							✓	✓		
ValueStep	Double							✓	✓		
Width	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

	Wert	UnoControlRadioButtonModel	UnoControlCheckBoxModel	UnoControlImageControlModel	UnoControlListBoxModel	UnoControlComboBoxModel	UnoControlScrollBarModel	UnoControlGroupBoxModel	UnoControlFixedLineModel	UnoControlProgressBarModel	UnoControlFileControlModel
Align	Integer										
AutoComplete	Boolean					✓					
AvailableServiceNames	Array										
BackgroundColor	Long				✓	✓				✓	✓
BlockIncrement	Long						✓				
Border	Integer			✓	✓	✓	✓			✓	✓
Closeable	Boolean										
ControlModels	Array										
CurrencySymbol	String										
Date	Long										
DateFormat	Integer										
DateMax	Long										
DateMin	Long										
DateShowCentury	Boolean										
DecimalAccuracy	Integer										
DefaultControl	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DefaultButton	Boolean										
Delegator	Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Dropdown	Boolean				✓	✓					
EchoChar	Integer										
EditMask	String										
EffectiveDefault	Variant										
EffectiveMax	Double										
EffectiveMin	Double										
EffectiveValue	Variant										

	Wert	UnoControlRadioButtonModel	UnoControlCheckBoxModel	UnoControlImageControlModel	UnoControlListBoxModel	UnoControlComboBoxModel	UnoControlScrollBarModel	UnoControlGroupBoxModel	UnoControlFixedLineModel	UnoControlProgressBarModel	UnoControlFileControlModel
ElementNames	Array										
ElementType	Object										
Enabled	Boolean	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Events	Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FillColor	Long									✓	
FontCharset	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontCharWidth	Single	✓	✓	✓	✓	✓		✓	✓		✓
FontDescriptor	Object	✓	✓	✓	✓	✓		✓	✓		✓
FontEmphasisMark	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontFamily	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontHeight	Single	✓	✓	✓	✓	✓		✓	✓		✓
FontKerning	Boolean	✓	✓	✓	✓	✓		✓	✓		✓
FontName	String	✓	✓	✓	✓	✓		✓	✓		✓
FontOrientation	Single	✓	✓	✓	✓	✓		✓	✓		✓
FontPitch	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontRelief	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontSlant	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontStrikeout	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontStyleName	String	✓	✓	✓	✓	✓		✓	✓		✓
FontType	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontUnderline	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontWeight	Single	✓	✓	✓	✓	✓		✓	✓		✓
FontWidth	Integer	✓	✓	✓	✓	✓		✓	✓		✓
FontWordLineMode	Boolean	✓	✓	✓	✓	✓		✓	✓		✓
FormatKey	Long										

	Wert	UnoControlRadioButtonModel	UnoControlCheckBoxModel	UnoControlImageControlModel	UnoControlListBoxModel	UnoControlComboBoxModel	UnoControlScrollBarModel	UnoControlGroupBoxModel	UnoControlFixedLineModel	UnoControlProgressBarModel	UnoControlFileControlModel
FormatSupplier	Object										
GroupControl	Boolean										
GroupCount	Long										
Height	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HelpText	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HelpURL	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HardLineBreaks	Boolean										
HScroll	Boolean										
ImageAlign	Integer										
ImageURL	String			✓							
ImplementationId	Array	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ImplementationName	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Label	String	✓	✓					✓	✓		
LineCount	Integer				✓	✓					
LineIncrement	Long						✓				
LiteralMask	String										
MaxTextLen	Integer					✓					
Moveable	Boolean										
MultiLine	Boolean										
MultiSelection	Boolean				✓						
Name	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Orientation	Long						✓		✓		
PositionX	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PositionY	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PrependCurrencySymbol	Boolean										

	Wert	UnoControlRadioButtonModel	UnoControlCheckBoxModel	UnoControlImageControlModel	UnoControlListBoxModel	UnoControlComboBoxModel	UnoControlScrollBarModel	UnoControlGroupBoxModel	UnoControlFixedLineModel	UnoControlProgressBarModel	UnoControlFileControlModel
Printable	Boolean	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ProgressValue	Long									✓	
ProgressValueMax	Long									✓	
ProgressValueMin	Long									✓	
PropertySetInfo	Object	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PropertyToDefault	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PushButtonType	Integer										
ReadOnly	Boolean				✓	✓					✓
ScaleImage	Boolean			✓							
ScrollValue	Long						✓				
SelectedItems	Array				✓						
ServiceName	String	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ShowThousendsSeperator	Boolean										
Sizeable	Booelan										
Spin	Boolean										
State	Integer	✓	✓								
Step	Long	✓	✓	✓	✓		✓	✓	✓	✓	✓
StrictFormat	Boolean										
StringItemList	Array				✓	✓					
SupportedServiceNames	Array	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TabIndex	Integer	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TabStop	Boolean	✓	✓	✓	✓	✓					✓
Tag	String	✓	✓	✓	✓		✓	✓	✓	✓	✓
Text	String					✓					✓
TextColor	Long	✓	✓	✓	✓	✓		✓	✓		✓

	Wert	UnoControlRadioButtonModel	UnoControlCheckBoxModel	UnoControlImageControlModel	UnoControlListBoxModel	UnoControlComboBoxModel	UnoControlScrollBarModel	UnoControlGroupBoxModel	UnoControlFixedLineModel	UnoControlProgressBarModel	UnoControlFileControlModel
TextLineColor	Long	✓	✓	✓	✓	✓		✓	✓		✓
Time	Long										
TimeFormat	Integer										
TimeMax	Long										
TimeMin	Long										
Title	String										
TreatAsNumber	Boolean										
TriState	Boolaen		✓								
Types	Array	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VScroll	Boolean										
Value	Double										
ValueMax	Double										
ValueMin	Double										
ValueStep	Double										
VisibleSize	Long						✓				
Width	Long	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Dialoge

In diesem Kapitel wird der Aufbau der Befehle von OOo erläutert – es wird gezeigt, wie man sich die Eigenschaften und Methoden auflisten kann und wie sämtliche Befehle strukturiert sind.

3 Die StarOffice API

3.1 UNO (Universal Network Objects)

Die StarOffice API (Application Programming Interface) ist das Komponentenmodell, auf dem die Applikation StarOffice basiert und mit deren Hilfe sie automatisiert wird. Es stellt eine Definition von allen Objekten und Interfaces zur Verfügung, die angesteuert werden können. Sie kann innerhalb des Programms von OOo mit Basic, Java oder Python, aber auch von außen, das heißt von Java, VS.NET oder C++ angesteuert werden. Dies wird durch die Technologie mit dem Namen UNO (Universal Network Objects) möglich gemacht. Es ist ein Komponentenmodell, das die Interoperabilität zwischen den einzelnen Elementen (Objekten, Methoden, Prozessen) und den Programmiersprachen zur Verfügung stellt.

3.1.1 Services

Ein Service ist – anders als Objekte – nicht direkt aufrufbar, sondern man muss, um UNO in OOo zu verwenden, eine Variable vom Typ `Object` deklarieren:

```
Dim Obj As Object
```

Bedauerlicherweise existieren in OOo nur Objekte von Typ `Object` – anders als in anderen Programmiersprachen, wo die Objekte differenzierter dargestellt werden.

Einige Objekte stehen direkt zum Zugriff zur Verfügung, beispielsweise in Calc. Dort kann in die erste Zelle links oben (A1) ein Text geschrieben werden:

```
Dim objDatei as Object
Dim objBlaetter As Object
Dim objBlatt As Object
Dim objZelle As Object
objDatei = ThisComponent
objBlaetter = objDatei.Sheets
objBlatt = objBlaetter.getByIndex(0)
objZelle = objBlatt.getCellByPosition(0, 0)
objZelle.String = "Mensch & Objekt"
```

Man könnte es auch in eine Zeile schreiben, was allerdings nicht gerade die Lesbarkeit und Wartbarkeit des Codes erhöht:

```
ThisComponent.Sheets.getByIndex(0).getCellByPosition(0, 0).String = _
   "Mensch & Objekt"
```

Mir ist aufgefallen, dass an einigen Stellen OOo Probleme mit solchen „langen Kaskaden“ hat – aus vielerlei Gründen ist es vernünftig, diesen Befehl auf mehrere Zeilen, das heißt auf mehrere Objekte, Eigenschaften und Methoden zu verteilen.

Andere Variablen müssen initialisiert werden, das bedeutet: Sie stellen einen Verweis auf den gewünschten Service her:

```
Obj = createUnoService("com.sun.star.frame.Desktop")
```

Über diesen Aufruf wird beispielsweise ein Verweis auf die OOo-Applikation hergestellt – in der UNO-Terminologie wird von Service gesprochen.

Tabelle 3.33 Folgende Funktionen werden von UNO zur Verfügung gestellt

Funktion	**Beschreibung**
CreateUnoDialog()	erstellt einen Dialog, der bereits erzeugt wurde (Kapitel 2)
CreateUnoListener()	greift auf die Ereignisse zu
CreateUnoService()	erstellt einen UNO-Service
CreateUnoStruct()	erstellt eine Struktur (Kapitel 3.1.7)
CreateUnoValue()	erstellt einen UNO-Value
DialogLibraries	greift auf die Bibliotheken zu
EqualUNOObjects()	vergleicht zwei UNO-Objekte
FindObject()	findet ein Objekt
FindObjectProperty()	findet eine Objekteigenschaft
GetDefaultContext()	erstellt eine Kopie des Kontextes
GetProcessServiceManager()	greift auf den Service-Manager zu
GlobalScope	die Bibliotheken der OOo-Umgebung
HasUnoInterfaces()	überprüft, ob ein Objekt, das eine Schnittstelle unterstützt
IsUnoStruct()	überprüft, ob eine Variable ein UNO-Objekt darstellt
ThisComponent	Objekt, das auf das Dokument zugreift, an dem der Code hängt

OOo stellt eine Reihe an Services zur Verfügung, die in Modulen zusammengefasst wurden. Für die Verwendung ist die Angabe des Modulnamens wichtig, da er mit im Namen aufgeführt werden muss.

Der vollständige Name eines Services setzt sich immer zusammen aus dem Ausdruck „com.sun.star“, der angibt, dass es sich um einen OOo-Service handelt (andere gibt es nicht), und aus dem Modulnamen, beispielsweise „frame“, und dem eigentlichen Servicenamen, wie hier zum Beispiel „Desktop“.

Tabelle 3.34 Folgende Services stehen zur Zeit zur Verfügung

Servicename	**Beschreibung**
accessibility	UNO-ZUgriff
auth	„authenticates interfaces“ (Sicherheit)
awt	Benutzerschnittstelle für UNO.
beans	Java beans-ähnliche Eigenschaft für den Zugriff
bridge	Schnittstelle für Bridges zu anderen Komponenten
chart	Diagramme
configuration	Daten-Konfiguration
connection	Austausch
container	Schnittstelle für Sammlungen
corba	Module für IOP, IIOP and GIOP
datatransfer	Datenaustausch (Zwischenspeicher und Drag & Drop)
document	Textdokumente
drawing	Zeichnungsdokumente
form	Schnittstelle für Formulare und Steuerelemente
formula	Formeldokumente
frame	der „Rahmen“ von OOo
image	Bildbearbeitung
installation	Schnittstelle für Installationen
io	allgemeine Input/Output Schnittstelle.
java	Java to UNO connectivity-Schnittstelle
lang	allgemeine UNO-Konzepte
ldap	Lightweight Directory Access Protocol (LDAP)
linguistic2	Rechtschreibprüfung, Silbentrennung und Thesaurus
loader	Schnittstelle für den „Loader“
mozilla	Schnittstelle für die Mozilla-Integration
package	Schnittstelle für verschiedene Dateiformate
packages	Schnittstelle für die Sammlung an Dateiformaten
plugin	Schnittstelle für ein Netscape-ähnliches Plugin
presentation	Präsentationsdokument
reflection	„Runtime Object Inspection/Core Reflection Interfaces“
registry	Schnittstelle um Datenbanken zu registrieren
resource	Schnittstelle, um auf „Resource Files“ zuzugreifen (UI)

Servicename	Beschreibung
scanner	Schnittstelle für Scanner
script	Scripting Language
sdb	Schnittstelle für Datenbankzugriff
sdbc	Schnittstelle für Datenbankkomponenten
sdbcx	High-Level Database Component Interfaces
security	Schnittstelle für „Authorisation“ und „Authentification“
setup	Schnittstelle, um Setup-Prozesse zu kontrollieren
sheet	Tabellendokumente
style	Syles (Vorlagen)
svg	Schnittstelle für Scaleable Vector Format (SVG)
sync2	Schnittstelle zur Datensynchronisierung
system	Schnittstelle zur Integration von Host Operating Systems
table	Tabellen in Text- und Tabellendokumenten
task	Schnittstelle zum Task Management
test	Schnittstelle für „Core-Reflection“, „Introspection“ und „Marshalling“
text	Schnittstelle für Texte
ucb	Schnittstelle für „Universal Content Broker“
ui	Dialoge und Kontrollelemente
uno	Basic UNO
util	Vermischte Schnittstellen zum Sortieren, Objekte verbinden …
view	Druckvorschau
xml	Schnittstelle für XML

Neben den Begriffen Modul und Service führt UNO den Begriff der Schnittstelle (Interface) ein. Ein Interface fasst mehrere Methoden zusammen. Streng genommen unterstützt ein Service in UNO keine Methoden, sondern Schnittstellen, die wiederum verschiedene Methoden bereitstellen. Anders ausgedrückt: Die Methoden werden dem Service in Schnittstellen zusammengefasst zugeordnet. Denn dort wird die Schnittstelle für den Aufruf einer Methode benötigt. In OOo-Basic ist dies irrelevant. Hier werden die Methoden direkt über das jeweilige Objekt aufgerufen.

Um neue Objekte zu erstellen, muss die Methode `createInstance` mit dem entsprechenden Service auf das „übergeordnete“ Objekt angewendet werden. Das folgende Beispiel erzeugt ein neues Tabellenblatt in einer Calc-Datei:

```
Dim objDatei As Object
```

```
Dim objBlatt As Object

objDatei = ThisComponent
objBlatt = ThisComponent.Sheets(0)
If objDatei.Sheets.hasByName("MeinBlatt") Then
   objBlatt = objDatei.Sheets.getByName("MeinBlatt")
Else
   objBlatt = objDatei.createInstance _
      ("com.sun.star.sheet.Spreadsheet")
   objDatei.Sheets.insertByName("MeinBlatt", objBlatt)
End If
```

3.1.2 Die Dokumentation der Eigenschaften, Methoden und Schnittstellen

Nun stellt sich die Frage, welche Objekte (Services) welche Eigenschaften, Methoden und Schnittstellen unterstützen und wie sich diese ermitteln lassen. Es existieren drei Hilfen:

- Der Developer's Guide der API-Referenz
- Die Debug-Methoden
- Die Methode supportsService

API-Referenz

Sämtliche Informationen über die vorhandenen Services, ihre Schnittstellen, Methoden und Eigenschaften mit Kommentaren und Erläuterungen liefert die API-Referenz für das OOo API. Sie steht im Internet zur Verfügung unter:

http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html

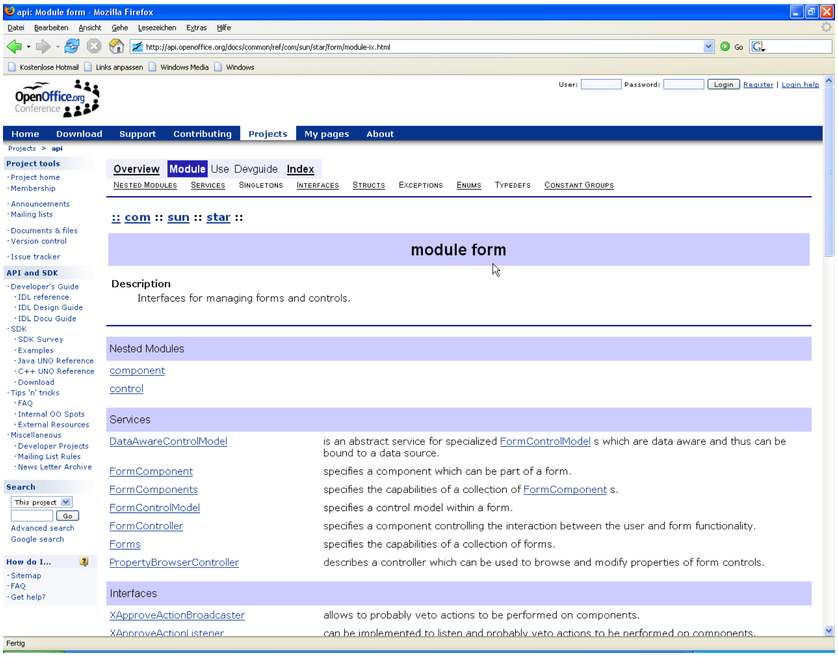


Abbildung 3.63 Ein Objekt mit Services, Interfaces, Structs, Exceptions, Enums und Konstanten

Die Debug-Eigenschaften

Jedes UNO-Objekt liefert Informationen darüber, welche Eigenschaften, Methoden und Schnittstellen es zur Verfügung stellt. Es bietet Eigenschaften, die diese in Form einer Liste ausgeben.

Tabelle 3.35 Die entsprechenden Eigenschaften der Objekte lauten:

Eigenschaften	Beschreibung
DBG_properties	liefert einen String mit allen Eigenschaften eines Objektes
DBG_methods	liefert einen String mit allen Methoden eines Objektes
DBG_supportedInterfaces	liefert einen String mit allen Interfaces, die ein Objekt unterstützt. Sie beginnen immer mit dem Buchstaben „X“.

Das nächste Beispiel zeigt, wie sich `DBG_properties` und `DBG_methods` verwenden lassen. Zuerst wird der Service „com.sun.star.frame.Desktop“ erzeugt, der an die Objektvariable `Obj` übergeben wird. Anschließend werden die unterstützten Eigenschaften und Methoden in Meldungsfenstern ausgegeben:

```
Dim Obj As Object
Obj = createUnoService("com.sun.star.frame.Desktop")
MsgBox Obj.DBG_properties
MsgBox Obj.DBG_methods
```

Wenn Sie sich die Liste der Eigenschaften anzeigen lassen, dann liefert `createUnoService("com.sun.star.frame.Desktop").DBG_Properties` oder `StarDesktop.DBG_Properties`:

```
Properties of object "com.sun.star.frame.Desktop":
SbxOBJECT ActiveFrame
SbxOBJECT DispatchRecorderSupplier
SbxBOOL IsPlugged
SbxBOOL SuspendQuickstartVeto
SbxSTRING Title
SbxARRAY Types
SbxARRAY ImplementationId
SbxSTRING ImplementationName
SbxARRAY SupportedServiceNames
SbxOBJECT Components
SbxOBJECT CurrentComponent
SbxOBJECT CurrentFrame
SbxOBJECT Tasks
SbxOBJECT ActiveTask
SbxOBJECT ContainerWindow
SbxOBJECT Creator
SbxSTRING Name
SbxOBJECT ComponentWindow
SbxOBJECT Controller
SbxOBJECT Frames
SbxOBJECT PropertySetInfo
SbxSTRING Dbg_SupportedInterfaces
SbxSTRING Dbg_Properties
SbxSTRING Dbg_Methods
```

Sie können diese Liste in ein neues Textdokument (oder alternativ in eine Tabellenzelle oder ein Shape in einer Draw-Zeichnung) schreiben lassen:

```
Sub Methoden_und_EigenschaftenAuflisten
   Dim objDoc As Object
    Dim objCursor As Object
   Dim docArgs() As New com.sun.star.beans.PropertyValue
   Dim strUrl As String

   strUrl = "private:factory/swriter"
   objDoc = StarDesktop.loadComponentFromURL _
      (strUrl, "_blank", 0, docArgs())
    objCursor = objDoc.Text.createTextCursor()

   objCursor.String = StarDesktop.DBG_Properties
   objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, 0, False)
   objDoc.Text.insertControlCharacter(objCursor, 0, False)
   objCursor.gotoEnd(False)
   objCursor.String = StarDesktop.DBG_Methods

End Sub
```

Dabei können die Strings, Boolean-Werte und Zahlen direkt ausgegeben werden:

```
   Dim Obj As Object

   Obj = createUnoService("com.sun.star.frame.Desktop")

   MsgBox Obj.IsPlugged
   MsgBox Obj.SuspendQuickstartVeto
   MsgBox Obj.Title
   MsgBox Obj.ImplementationName
   MsgBox Obj.Name
```

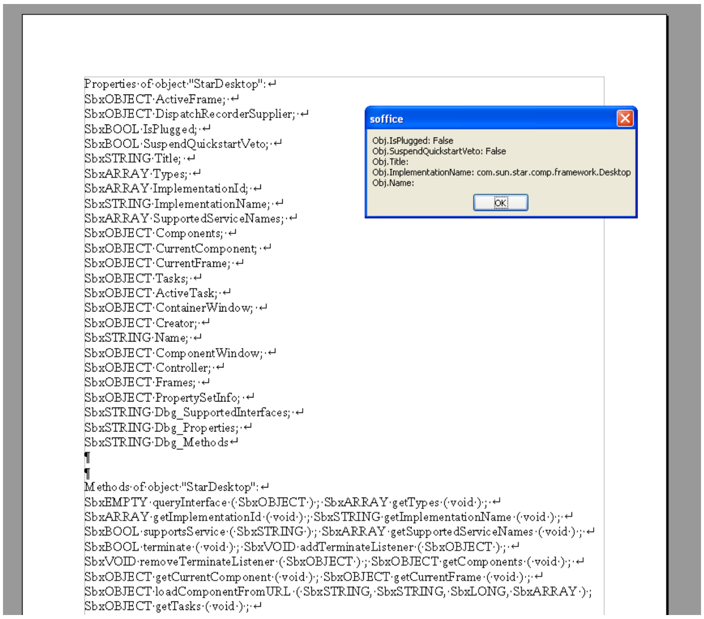


Abbildung 3.64 Das neue Dokument mit den Methoden und Eigenschaften und ein Meldungsfenster mit einigen Eigenschaften

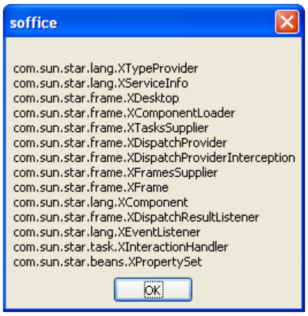


Abbildung 3.65 Ein Datenfeld mit seinen Elementen

Die Arrays können durchlaufen werden:

```
Dim Obj As Object
Dim i As Integer
Dim strListe As String
```

```
Obj = createUnoService("com.sun.star.frame.Desktop")
For i = 0 To Ubound(Obj.Types)
   strListe = strListe & Chr(13) & Obj.Types(i).Name
Next i
MsgBox strListe
```

Und auf Objekte kann ebenfalls zugegriffen werden (siehe Abbildung 3.4):

```
Dim Obj As Object
Dim ObjTiefer As Object
Obj = createUnoService("com.sun.star.frame.Desktop")
ObjTiefer = Obj.ActiveFrame
MsgBox ObjTiefer.DBG_Properties
MsgBox ObjTiefer.DBG_methods
MsgBox ObjTiefer.DBG_supportedInterfaces
```

Bei der Verwendung der Eigenschaft `DBG_properties` ist zu beachten, dass sie alle Eigenschaften zurückliefert, die ein bestimmter Service theoretisch unterstützen kann. Es ist jedoch nicht sichergestellt, dass diese von dem betreffenden Objekt auch verwendet werden. Vor dem Auslesen einer Eigenschaft kann mit der Funktion `IsEmpty` geprüft werden, ob diese tatsächlich verfügbar ist.

3.1.3 Eigenschaften von Objekten

Eigenschaften (Properties) geben Werte von folgendem Typ zurück:

`Boolean`, `Integer`, `Long`, `Short`, `Double`, `String`, `Variant`, `Object` und `Array`

```
Obj = createUnoService("com.sun.star.frame.Desktop")
MsgBox Obj.DBG_properties
MsgBox Obj.ImplementationName
```

Alle Eigenschaften können abgefragt werden, nicht alle können verändert werden, das heißt, eine Reihe von Eigenschaften ist schreibgeschützt. Folgendes Beispiel liefert einen Fehler:

```
Obj = createUnoService("com.sun.star.frame.Desktop")
MsgBox Obj.DBG_properties
Obj.ImplementationName = "com.sun.star.frame.WasWeissIch"
```

Viele Eigenschaften können über zweierlei Arten geschrieben werden:

```
objBlatt.Columns(5).setPropertyValue("Width", 6000)
objBlatt.Columns(5).Width = 6000
```

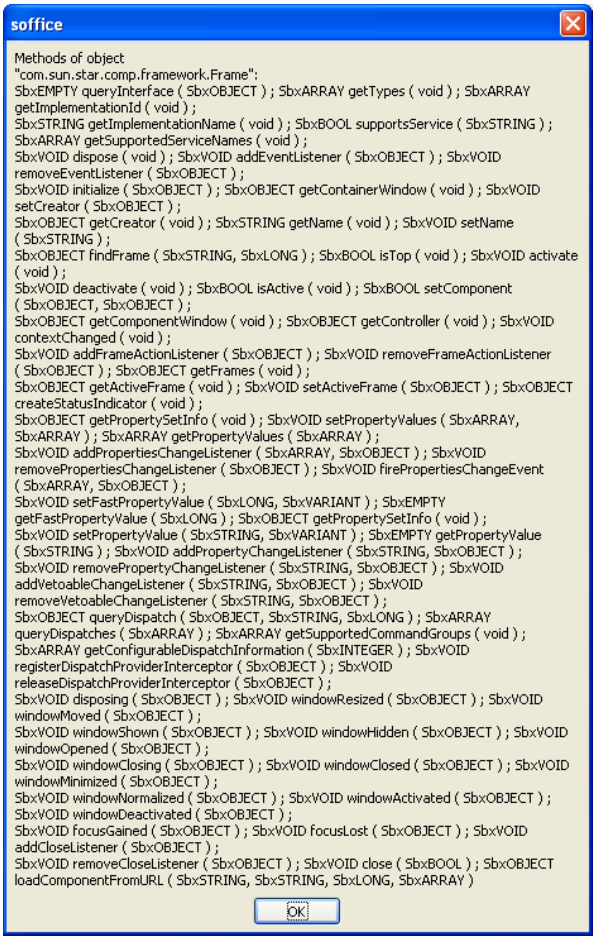


Abbildung 3.66 Die Methoden des Frame

Analog das Lesen der Eigenschaften:

```
MsgBox objBlatt.Columns(5).getPropertyValue("Width")
MsgBox objBlatt.Columns(5).Width
```

In diesem Buch habe ich mich weitgehend auf die zweite Schreibweise beschränkt.

3.1.4 Methoden von Objekten

Analog zum obigen Beispiel liefert `DBG_methods`:

```
SbxEMPTY queryInterface ( SbxOBJECT )
SbxARRAY getTypes ( void )
SbxARRAY getImplementationId ( void )
SbxSTRING getImplementationName ( void )
SbxBOOL supportsService ( SbxSTRING )
SbxARRAY getSupportedServiceNames ( void )
SbxBOOL terminate ( void )
SbxVOID addTerminateListener ( SbxOBJECT )
SbxVOID removeTerminateListener ( SbxOBJECT )
SbxOBJECT getComponents ( void )
SbxOBJECT getCurrentComponent ( void )
SbxOBJECT getCurrentFrame ( void )
SbxOBJECT loadComponentFromURL (SbxSTRING, SbxSTRING, SbxLONG, SbxARRAY)
SbxOBJECT getTasks ( void )
SbxOBJECT getActiveTask ( void )
SbxOBJECT queryDispatch ( SbxOBJECT, SbxSTRING, SbxLONG )
SbxARRAY queryDispatches ( SbxARRAY )
SbxVOID dispose ( void )
SbxVOID addEventListener ( SbxOBJECT )
SbxVOID removeEventListener ( SbxOBJECT )
SbxVOID initialize ( SbxOBJECT )
SbxOBJECT getContainerWindow ( void )
SbxVOID setCreator ( SbxOBJECT )
SbxOBJECT getCreator ( void )
SbxSTRING getName ( void )
SbxVOID setName ( SbxSTRING )
SbxOBJECT findFrame ( SbxSTRING, SbxLONG )
SbxBOOL isTop ( void )
SbxVOID activate ( void )
```

```
SbxVOID deactivate ( void )
SbxBOOL isActive ( void )
SbxBOOL setComponent ( SbxOBJECT, SbxOBJECT )
SbxOBJECT getComponentWindow ( void )
SbxOBJECT getController ( void )
SbxVOID contextChanged ( void )
SbxVOID addFrameActionListener ( SbxOBJECT )
SbxVOID removeFrameActionListener ( SbxOBJECT )
SbxOBJECT getFrames ( void )
SbxOBJECT getActiveFrame ( void )
SbxVOID setActiveFrame ( SbxOBJECT )
SbxVOID disposing ( SbxOBJECT )
SbxVOID dispatchFinished ( SbxOBJECT )
SbxVOID handle ( SbxOBJECT )
SbxOBJECT getPropertySetInfo ( void )
SbxVOID setPropertyValue ( SbxSTRING, SbxVARIANT )
SbxEMPTY getPropertyValue ( SbxSTRING )
SbxVOID addPropertyChangeListener ( SbxSTRING, SbxOBJECT )
SbxVOID removePropertyChangeListener ( SbxSTRING, SbxOBJECT )
SbxVOID addVetoableChangeListener ( SbxSTRING, SbxOBJECT )
SbxVOID removeVetoableChangeListener ( SbxSTRING, SbxOBJECT )
```

Methoden können Parameter von folgenden Werten aufnehmen:

`Void, Boolean, Integer, Long, Short, Double, String, Variant, Object` und `Array`

Methoden nehmen keinen Wert auf (`Void`), einen Parameter oder mehrere Parameter. Sie können keinen Wert zurückgeben (`Void`) oder einen Rückgabewert liefern, der nicht notwendigerweise weiter verarbeitet werden muss.

Dies soll an einem Beispiel gezeigt werden: Mit dem Objekt `Obj` wird auf den Desktop zugegriffen:

```
Obj = createUnoService("com.sun.star.frame.Desktop")
```

Die Auflistung der Methoden `Obj.Dbg_methods` liefert die Methode `loadComponentFromURL` mit:

```
SbxOBJECT loadComponentFromURL ( SbxSTRING, SbxSTRING, SbxLONG, SbxARRAY)
```

Welcher Parameter hierfür verlangt wird, verrät die Internetseite (http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html):

```
::com::sun::star::lang::XComponent

loadComponentFromURL(

[in] string (URL)

[in] string (TargetFrameName)

[in] long (SearchFlags)

[in] sequence< ::com::sun::star::beans::PropertyValue >
```

Die Liste der „PropertyValue“ finden Sie von dieser Seite über einen Hyperlink:

```
AsTemplate, Author, CharacterSet, Comment, ComponentData, DocumentTitle, FileName,
FilterName, FilterFlags, FilterOptions, FilterData, Hidden, InputStream, Interac-
tionHandler, JumpMark, MediaType, OpenFlags, OpenNewView, Overwrite, Password,
PostData, PostString, Preview, ReadOnly, StartPresentation, Referer, RepairPacka-
ge, StatusIndicator, TemplateName, TemplateRegionName, Unpacked, URL, Version,
ViewData, ViewId, MacroExecutionMode und UpdateDocMode
```

Sie sind wiederum kommentiert. Das bedeutet, eine Datei wird wie folgt konvertiert und geöffnet:

```
strURL = "C:\kap01.doc"

strURL = ConvertToUrl(strURL)

docArgs(0).Name = "FilterName"

docArgs(0).Value = "MS Word"

docArgs(1).Name = "AsTemplate"

docArgs(0).Value = True

ObjDok = Obj.loadComponentFromURL(strURL, "_blank", 0, docArgs())
```

Beachten Sie, dass Sie eine Reihe von Attributen sowohl als Eigenschaft als auch über eine Methode erfragen und setzen können – sie liegen häufig als Dublette vor. Im nächsten Beispiel werden die Eigenschaft `Name` und die Methode `getName` verwendet, um den Namen des ersten Blattes einer Calc-Datei zu ermitteln:

```
Dim objBlaetter As Object

objBlaetter = ThisComponent.Sheets

MsgBox objBlaetter.getByIndex(0).Name

MsgBox objBlaetter.getByIndex(0).getName()
```

Umgekehrt kann das erste Blatt über die Eigenschaft `Name` und ebenso über die Methode `setName(String)` umbenannt werden:

```
Dim objBlaetter As Object
```

```
objBlaetter = ThisComponent.Sheets
objBlaetter.getByIndex(0).Name = "Sheet1"
MsgBox "umbenannt"
objBlaetter.getByIndex(0).setName("Tabelle1")
```

Das Meldungsfenster dazwischen dient lediglich dazu, um das Umbenennen zu verdeutlichen.

3.1.5 Sammlungen

Eine Reihe von Objekten wird in Sammlungen zusammengefasst. Sehr viele Sammlungen verfügen über die Eigenschaft `getElementNames`, mit der auf sämtliche Namen zugegriffen werden kann. Es gibt einen Array zurück, der durchlaufen werden kann. Das nächste Beispiel zeigt, wie alle Blattnamen einer Calc-Datei ausgelesen werden können:

```
Sub Blatt_Zugriff

   Dim objBlaetter As Object
   Dim strBlattNamen() As String
   Dim strBlattListe As String
   Dim i As Integer

   objBlaetter = ThisComponent.Sheets
   strBlattNamen = objBlaetter.getElementNames
   For i = LBound(strBlattNamen) To UBound(strBlattNamen)
      strBlattListe = strBlattListe & Chr(13) & strBlattNamen(i)
   Next i

   MsgBox strBlattListe
End Sub
```

Der Zugriff funktioniert auch mit einer Zählervariablen und der Eigenschaft `getByIndex`:

```
For i = 0 to objBlaetter.getCount() - 1
   objBlatt = objBlaetter.getByIndex(i)
   strBlattListe =  strBlattListe & Chr(13) & objBlatt.Name
Next i
```

```
    MsgBox strBlattListe
```

Die Anzahl von fast jeder Sammlung kann sowohl mit der Eigenschaft `Count` als auch mit der Methode `getCount()` ermittelt werden.

Beachten Sie, dass (fast) alle Sammlungen beim Index 0 beginnen, also: „Tabelle1“ hat den Index 0, die Zelle A1 ist die Spalte Nr. 0 und die Zeile Nr. 0 (`getCellByPosition(0, 0)`). Die einzige Ausnahme liegt bei den Datenbanken vor: Dort beginnt die Zählung bei 1!

Mit der Methode `CreateEnumeration` wird ein Aufzählungsobjekt erstellt, das mit der Methode `nextElement()` von vorne nach hinten durchlaufen werden kann. Jedes Objekt kann an eine Objektvariable übergeben und weiter verarbeitet werden. Um zu überprüfen, ob noch weitere Objekte vorhanden sind, steht die Methode `hasMoreElements()` zur Verfügung:

```
    objBlaetter = ThisComponent.Sheets.CreateEnumeration
    Do
        objBlatt = objBlaetter.nextElement
        strBlattListe =  strBlattListe & Chr(13) & objBlatt.Name
    Loop While objBlaetter.hasMoreElements()

    MsgBox strBlattListe
```

Die Methode `hasByName(String)` verrät, ob ein Tabellenblatt mit einem bestimmten Namen innerhalb der Datei vorhanden ist.

Die Methode `hasByName` existiert in den meisten Sammlungen und sollte selbstverständlich eingesetzt werden, um Fehler abzufangen!

So gibt das folgende Beispiel eine Meldung darüber aus, ob das Sammlung „objBlaetter“ ein Tabellenblatt mit dem Namen „Tabelle99“ enthält.

```
Sub Blatt_Zugriff03
    Dim objBlaetter As Object
    objBlaetter = ThisComponent.Sheets
    If objBlaetter.HasByName("Tabelle99") Then
        MsgBox "Das Blatt ""Tabelle99"" ist vorhanden."
    Else
        MsgBox "Das Blatt ""Tabelle99"" ist nicht vorhanden."
    End If
End Sub
```

Interessanterweise verfügen die meisten Sammlungen sowohl über die Eigenschaft `getByIndex` oder `getByName` – allerdings ist letztere oft nicht bekannt, deshalb muss über den Index iteriert werden.

3.1.6 Konstanten

Eine Reihe von Objekten besitzt Konstanten. Sie bestehen aus einer internen Zahl und einer Struktur in der Schreibweise com.sun.star.Servicename.Konstante. Beispielsweise besitzt die Eigenschaft CharWeight (Schriftstärke) in Writer die Konstanten gemäß com.sun.star.awt.FontWeight: DONTKNOW (0), THIN (50), ULTRALIGHT (60), LIGHT (75), SEMILIGHT (90), NORMAL (100), SEMIBOLD (110), BOLD (150), ULTRABOLD (175) und BLACK (200), also com.sun.star.awt.FontWeight.NORMAL oder den Wert 100. „Fett" entspricht folglich der Konstante „com.sun.star.awt.FontWeight.BOLD" oder 150. Im folgenden Beispiel wird die Schrift auf Arial 14 pt, unterstrichen, fett, blau und zentriert eingestellt, bevor der Text „geschrieben" wird:

```
objCursor = objDoc.Text.createTextCursor
objCursor.CharFontName = "Arial"
objCursor.CharHeight = 14
objCursor.CharUnderline = com.sun.star.awt.FontUnderline.SINGLE
objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
objCursor.CharColor = RGB(0, 0, 255)
objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.CENTER
objCursor.String = "Anstellungsvertrag"
```

Zur besseren Lesbarkeit des Codes sollten Sie den Konstantennamen ausschreiben.

Beachten Sie, dass die Konstantennamen exakte Groß- und Kleinschreibung verlangen!

Auch Farben liegen als Zahlen vor – sie können mit der Funktion `RGB` erzeugt werden.

Integrierte und benutzerdefinierte Zahlenformate liegen ebenfalls als (Integer-) Zahlen vor, wie auf sie zugegriffen werden kann, wird in Kapitel 5.4.6 erläutert.

3.1.7 Strukturen (structs)

Einige Objekte besitzen Strukturen („structs"). In einem Tabellenblatt kann auf die Zelle A1 wie folgt zugegriffen werden:

```
Sub Zellzugriff09

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objZelle As Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
   objZelle = objBlatt.getCellByPosition(0, 0)
```

In sie wird ein Text mit der Eigenschaft „String" geschrieben:

```
   objZelle.String = "Dies ist nur ein Test!"
```

Der Zellhintergrund wird mittels einer anderen Eigenschaft formatiert:

```
   objZelle.CellBackColor = RGB(255, 255, 0)
```

Um einen Schatten einzufügen, muss ein Objekt deklariert werden:

```
   Dim objSchattenFormat As New com.sun.star.table.ShadowFormat
```

Tabelle 3.36 ShadowFormat ist eine Struktur, die folgenden Aufbau besitzt:

Wert	Wert	Beschreibung
Location	Enum	Position des Schattens (Wert aus der Struktur com.sun.star.table.ShadowLocation: NONE, TOP_LEFT, TOP_RIGHT, BOTTOM_LEFT, BOTTOM_RIGHT)
ShadowWidth	Short	Größe des Schattens in 100stel Millimeter
IsTransparent	Boolean	setzt den Schatten transparent
Color	Long	Farbwert des Schattens

Das heißt, der Schatten der Zelle wird nun wie folgt formatiert:

```
   objSchattenFormat.Location = _
      com.sun.star.table.ShadowLocation.BOTTOM_RIGHT
   objSchattenFormat.ShadowWidth = 100
   objSchattenFormat.Color = RGB(0, 153, 255)
```

```
   objZelle.ShadowFormat = objSchattenFormat

End Sub
```

Ganz allgemein stellen Strukturen Wertepaare dar, die aus „Name“ und „Value“ bestehen, also etwa folgendermaßen:

```
   Dim objMitglied As New com.sun.star.beans.PropertyValue
   objMitglied.Name = "Vorname"
   objMitglied.Value = "Rene"

   MsgBox objMitglied.Value
```

Viele UNO-Strukturen bestehen jedoch aus Datenfeldern:

```
Sub Struktur2

   Dim objMitglied(9) As New com.sun.star.beans.PropertyValue
   objMitglied(0).Name = "Vorname"
   objMitglied(0).Value = "Rene"
   objMitglied(1).Name = "Vorname"
   objMitglied(1).Value = "Robert"
   objMitglied(2).Name = "Vorname"
   objMitglied(2).Value = "Ariane"

   MsgBox objMitglied(1).Value

End Sub
```

Soll das Objekt noch auf andere Art und Weise verwendet werden, kann man mit `CreateUnoStruct` arbeiten:

```
Sub Struktur3

   Dim objMitglied As Object
   objMitglied = CreateUnoStruct("com.sun.star.beans.PropertyValue")
   objMitglied.Name = "Vorname"
```

```
   objMitglied.Value = "Rene"

   MsgBox objMitglied.Value

End Sub
```

Konkret verwendet beispielsweise die bedingte Formatierung in Calc diese Name-Value-Struktur:

```
Sub Zellzugriff14

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objBereich As Object
   Dim objBedingung As Object
   Dim Bedingungen(2) As  new com.sun.star.beans.PropertyValue

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(3)
   objBereich = objBlatt.getCellRangeByName("A1:B10")

   Bedingungen(0).Name = "Operator"
   Bedingungen(0).Value = com.sun.star.sheet.ConditionOperator.GREATER
   Bedingungen(1).Name = "Formula1"
   Bedingungen(1).Value = "1"

   Bedingungen(2).Name = "StyleName"
   Bedingungen(2).Value = "Überschrift"

   objBedingung = objBereich.ConditionalFormat
   objBedingung.addNew(Bedingungen())

   objBereich.ConditionalFormat = objBedingung

End Sub
```

Es existieren zweierlei Arten von Strukturen. Während im oberen Beispiel sämtliche Name-Value-Paare optional sind, müssen im nächsten Beispiel alle fünf Werte angegeben werden, damit die Struktur „com.sun.star.table.CellRangeAddress“ korrekt beschrieben wird. Sie wird beispielsweise verwendet, um ein Diagramm zu erzeugen (wie im nächsten Beispiel) oder um einen Zellbereich zu löschen oder zu verschieben (s. Kapitel 5.3.2).

```
Sub Diagramm1

   Dim objDoc As Object
   Dim objDiagramme As Object
   Dim objDiaFlaeche As New com.sun.star.awt.Rectangle
   Dim objZellAdresse(0) As New com.sun.star.table.CellRangeAddress
   objDoc = ThisComponent
   objDiagramme = objDoc.Sheets(0).Charts

   objDiaFlaeche.X = 8000
   objDiaFlaeche.Y = 1000
   objDiaFlaeche.Width = 10000
   objDiaFlaeche.Height = 7000

   objZellAdresse(0).Sheet = 0
   objZellAdresse(0).StartColumn = 0
   objZellAdresse(0).StartRow = 0
   objZellAdresse(0).EndColumn = 1
   objZellAdresse(0).EndRow = 65

   objDiagramme.addNewByName _
      ("Diagramm1", objDiaFlaeche, objZellAdresse(), True, True)
End Sub
```

Beachten Sie, dass Sie zuerst die Strukturen generieren müssen und anschließend den entsprechenden Objekten zuweisen können. Im nächsten Beispiel wird eine Linie generiert, die an den unteren Rand einer Zelle „gebunden“ wird:

```
   Dim objLinie As New com.sun.star.table.BorderLine
```

```
   objLinie.Color = RGB(0, 0, 255)
   objLinie.OuterLineWidth = 10
   objZelle.BottomBorder = objLinie
```

Der umgekehrte Fall – erst die Zuweisung (`objZelle.BottomBorder = objLinie`) und anschließend die Definition `Color` und `OuterLineWidth` – würde zwar keinen Fehler erzeugen, aber auch nicht das Richtige tun – die Rahmenlinie wäre danach nicht formatiert.

Um also Strukturen korrekt einzusetzen, stehen Ihnen folgende zwei Möglichkeiten zur Verfügung. Entweder Sie deklarieren das Objekt als `Object` oder mit dem Schlüsselwort `New`.

Das folgende Beispiel erzeugt in Draw und Impress ein Rechteck. Das Objekt „objRechteck“ muss die beiden Strukturen „Size“ und „Position“ zugewiesen bekommen. Entweder werden sie so deklariert:

```
Sub Rechteck
   Dim objDoc As Object
   Dim objSeite As Object
   Dim objRechteck As Object
   Dim Punkt As New com.sun.star.awt.Point
   Dim Groesse As New com.sun.star.awt.Size
   objDoc = ThisComponent
   objSeite = objDoc.drawPages(0)
   Punkt.x = 10000:    Punkt.y = 5000
   Groesse.Width = 10000
   Groesse.Height = 10000
   objRechteck = _
      objDoc.createInstance("com.sun.star.drawing.RectangleShape")
   objRechteck.Size = Groesse
   objRechteck.Position = Punkt
   objSeite.add(objRechteck)
End Sub
```

Oder sie werden als Objekte deklariert und im Code als Struktur generiert:

```
Sub Rechteck2
   Dim objDoc As Object
   Dim objSeite As Object
   Dim objRechteck As Object
```

```
    Dim Punkt As Object
    Dim Groesse As Object
    objDoc = ThisComponent
    objSeite = objDoc.drawPages(0)
    Punkt = createUnoStruct("com.sun.star.awt.Point")
    Punkt.x = 10000:    Punkt.y = 5000
    Groesse = createUnoStruct("com.sun.star.awt.Size")
    Groesse.Width = 10000
    Groesse.Height = 10000
    objRechteck = _
       objDoc.createInstance("com.sun.star.drawing.RectangleShape")
    objRechteck.Size = Groesse
    objRechteck.Position = Punkt
    objSeite.add(objRechteck)
End Sub
```

Folgende Variante funktioniert nicht:

```
Dim Groesse As com.sun.star.awt.Size
```

Die Reihenfolge ist ebenfalls zwingend vorgeschrieben: Struktur erzeugen, Struktur füllen, Struktur(-objekt) einer Eigenschaft eines anderen Objektes übergeben.

In diesem Falle sieht das Rechteck wie folgt aus:

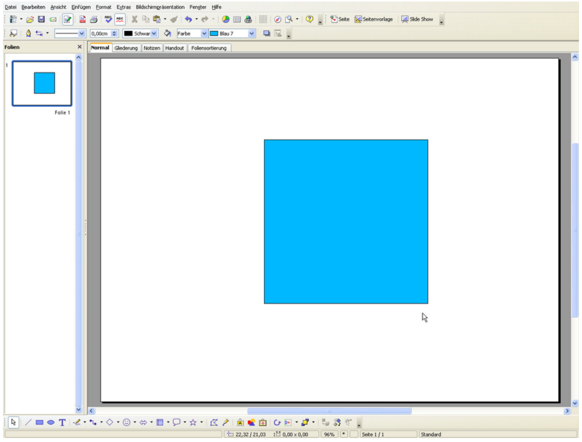


Abbildung 3.67 Das neue Rechteck

3.1.8 Die Methode supportsService

Mit der Methode `supportsService` kann überprüft werden, ob ein Objekt einen bestimmten Service unterstützt:

```
Dim Obj As Object

Obj = createUnoService("com.sun.star.frame.Desktop")

MsgBox Obj.supportsService("com.sun.star.frame.Desktop")
```

Somit funktioniert auch folgender Code:

```
Dim Doc As Object

Doc = StarDesktop.CurrentComponent

MsgBox Doc.supportsService("com.sun.star.frame.Desktop")
```

Er liefert „False“, da die aktuelle Komponente nicht den Service „com.sun.star.frame.Desktop“ unterstützt.

Die StarOffice API

In diesem Kapitel werden die Befehle erläutert, mit denen in OOo ein Dokument geöffnet oder ein neues erzeugt werden kann. Es wird beschrieben, wie Sie überprüfen können, in welcher Art Dokument sich der Benutzer befindet, wie man druckt und schließt.

4 Erzeugen kontextunabhängiger Objekte

Einige Services stehen nur in einer gewissen Umgebung (Textdokument, Tabellenblatt …) zur Verfügung, andere Services hängen nicht von einer entsprechenden Umgebung ab, sind also nicht an Writer, Calc … gebunden. Sie werden mit der Methode `createUnoService()` erstellt und sollen in diesem Kapitel beschrieben werden.

4.1 Der StarDesktop

Beim Umgang mit Dokumenten kommen im Wesentlichen zwei Services zum Einsatz:

Der zentrale Service „com.sun.star.frame.Desktop“ stellt die Funktionen für das Rahmenobjekt von OOo zur Verfügung, unter dem sich alle Dokumentfenster einreihen. Damit wird deutlich, dass Befehle, die der Benutzer über das Menü DATEI ausführt, über diesen Service ablaufen: das Erzeugen, Öffnen und Importieren von Dokumenten.

Die Basisfunktionalität für die einzelnen Dokument-Objekte liefert der Service „com.sun.star.document.OfficeDocument“. Er bietet die Methoden zum Speichern, Exportieren und Drucken von Dokumenten.

Der Service „com.sun.star.frame.Desktop“ steht mit dem Start von OOo automatisch zur Verfügung. Dazu erzeugt OOo ein Objekt, das über den globalen Namen StarDesktop erreichbar ist.

Die wichtigste Schnittstelle des Desktops von OpenOffice.org/StarOffice lautet „com.sun.star.frame.XComponentLoader“. Sie umfasst im Wesentlichen die Methode `loadComponentFromURL`, die für das Erzeugen, Importieren und Öffnen von Dokumenten zuständig ist.

4.2 Datei | Öffnen und Datei | Neu

Um ein Dokument zu öffnen, steht Ihnen der Service „Desktop“ zur Verfügung.

```
Dim ObjService As Object
Dim objDok As Object

ObjService = createUnoService("com.sun.star.frame.Desktop")
ObjDok = ObjService.loadComponentFromURL _
   ("file:///C:/Test.sxw", "_blank", 0, mNoArgs())
```

In OOo erfolgt der Zugriff auf das aktive Dokument-Objekt über die Eigenschaft `StarDesktop.CurrentComponent`.

Einige Informationen sind hierzu wichtig:

4.2.1 Dateinamen in URL-Notation

Da OOo als plattformunabhängige Anwendung konzipiert wurde, verwendet es für Dateinamen die betriebssystemunabhängige URL-Notation gemäß dem Internet-Standard RFC 1738.

Gewöhnliche Dateinamen beginnen dabei mit dem Präfix

```
file:///
```

gefolgt von dem lokalen Pfad. Enthält der Dateiname Unterverzeichnisse, so werden diese mit einfachen Schrägstrichen getrennt (also nicht mit dem unter Windows üblichen rückwärts gewandten Schrägstrich). Der folgende Pfad verweist auf die Datei TEST.SXW im Verzeichnis DOC auf dem Laufwerk C:

```
file:///C:/doc/test.sxw
```

Zur Umwandlung von lokalen Dateinamen in eine URL bietet OOo die Funktion `ConvertToUrl` an. Für die Rückumwandlung einer URL in einen lokalen Dateinamen hält OOo die Funktion `ConvertFromUrl` bereit:

```
MsgBox ConvertToUrl("C:\doc\test.sxw")
' liefert file:///C:/doc/test.sxw
MsgBox ConvertFromUrl("file:///C:/doc/test.sxw")
' liefert (unter Windows) c:\doc\test.sxw
```

Der zugrunde liegende Internet-Standard RFC 1738 gestattet die Verwendung der Zeichen 0 - 9, a - z und A - Z. Alle anderen Zeichen werden in einer Escape-Kodierung in die URLs eingefügt. Dazu werden sie in ihren Hexadezimalwert im Zeichensatz ISO 8859-1 (ISO-Latin) umgewandelt und bekommen ein Prozentzeichen vorangestellt. Aus einem Leerzeichen in einem lokalen Dateinamen wird so beispielsweise ein %20 in der URL.

4.2.2 Öffnen und Erstellen von OOo-Dokumenten

Das Öffnen und Bearbeiten von Dokumenten über die API erfolgt über den Desktop Service. Dieser Service bietet die Methode `loadComponentFromURL()` aus dem Interface `XComponentLoader` an. Durch die richtigen Argumente können bestimmte Eigenschaften des Dokumentes festgelegt werden:

```
loadCompontentFromURL( string aURL, string aTargetFrameName, long nSearchFlags,
sequence <com::sun::star::beans::PropertyValue> aArgs)
```

Tabelle 4.37 Die Parameter von loadCompontentFromURL

Argument	**Bedeutung**
aURL	URL des Dokuments, das geladen werden soll. Für neue Dokumente wird private:factory/scalc, private:factory/swriter, private:factory/simpress, private:factory/sdraw, private:factory/smath und private:factory/schart verwendet.
aTargetFrameName	Namen des Frames, in dem das Dokument geöffnet werden soll, _blank erzeugt einen neuen Frame _default ist speziell für UI-Funktionalität (beispielsweise um auf bereits geladene Dokumente zuzugreifen) _self der Rahmen selbst _parent das übergeordnete Element des Rahmens _top das oberste Element in der Hierarchie _beamer ein spezieller Unterrahmen
nSearchFlags	benutzt die Werte von FrameSearchFlag, um den entsprechenden TargetFrameName zu finden. Normalerweise wird 0 verwendet.
aArgs	optionale Eigenschaften: AsTemplate, Author, CharacterSet, Comment, ComponentData, DocumentTitle, FileName, FilterName, FilterFlags, FilterOptions, FilterData, Hidden, InputStream, InteractionHandler, JumpMark, MediaType, OpenFlags, OpenNewView, Overwrite, Password, PostData, PostString, Preview, ReadOnly, StartPresentation, Referer, RepairPackage, StatusIndicator, TemplateName, TemplateRegionName, Unpacked, URL, Version, ViewData, ViewId, MacroExecutionMode und UpdateDocMode Er wird als Variant-Array abgebildet.

Das erste Beispiel öffnet ein Dokument als Vorlage:

```
Sub API1
   Dim Obj As Object
   Dim objDok As Object
   Dim strURL As String
```

```
    Dim docArgs(1) As New com.sun.star.beans.PropertyValue

    Obj = createUnoService("com.sun.star.frame.Desktop")
    strURL = "C:\kap01.doc"
    strURL = ConvertToUrl(strURL)
    If Dir(strURL) <> "" Then
        docArgs(0).Name = "FilterName"
        docArgs(0).Value = "MS Word"
        docArgs(1).Name = "AsTemplate"
        docArgs(1).Value = True
        ObjDok = Obj.loadComponentFromURL(strURL, "_blank", 0, docArgs())
        MsgBox StarDesktop.CurrentComponent.URL
    Else
        MsgBox "Das Dokument wurde nicht gefunden."
    End If
End Sub
```

Übrigens kann auch statt

```
    Obj = createUnoService("com.sun.star.frame.Desktop")
```

geschrieben werden:

```
    Obj = StarDesktop.ActiveFrame
```

oder:

```
    Obj = StarDesktop
```

Das zweite Beispiel erzeugt ein leeres Textdokument:

```
Sub API2
    Dim docArgs()
    Dim strUrl As String
    Dim objDoc As Object
    strUrl = "private:factory/swriter"
    objDoc = StarDesktop.loadComponentFromURL(strUrl, "_blank", 0, _
        docArgs())
    MsgBox StarDesktop.CurrentComponent.URL
End Sub
```

Das dritte Beispiel öffnet ein Word-Dokument:

```
Sub Api3
   Dim ObjDesktop As Object
   Dim objDok As Object
   Dim strURL As String
   Dim mNoArgs()

   ObjDesktop = createUnoService("com.sun.star.frame.Desktop")
   strURL = "C:\kap01.doc"
   strURL = ConvertToUrl(strURL)

   If Dir(strURL) <> "" Then
      ObjDok = _
         ObjDesktop.loadComponentFromURL(strURL, "_blank", 0, mNoArgs())
      MsgBox StarDesktop.CurrentComponent.URL
   Else
      MsgBox "Das Dokument wurde nicht gefunden."
   End If
End Sub
```

4.2.3 nSearchFlags

Die genannten Werte `_blank` und `_hidden` für den Parameter „Frame" sorgen dafür, dass OOo für jeden Aufruf von `loadComponentFromURL` ein neues Fenster erzeugt. In einigen Situationen ist es jedoch sinnvoll, den Inhalt eines vorhandenen Fensters auszutauschen. In diesem Fall sollte das Rahmenobjekt des Fensters einen expliziten Namen erhalten (darf nicht mit Unterstrich beginnen). Des Weiteren muss der Parameter „SearchFlags" so gesetzt sein, dass der betreffende Rahmen erzeugt wird, falls er noch nicht vorhanden ist. Die betreffende Konstante für „SearchFlags" lautet:

```
lngSearchFlags = com.sun.star.frame.FrameSearchFlag.CREATE + _
com.sun.star.frame.FrameSearchFlag.ALL
```

Das nächste Beispiel zeigt, wie sich mit Hilfe der Parameter „Frame" und „SearchFlags" der Inhalt eines geöffneten Fensters austauschen lässt:

```
   objDim Doc As Object
```

```
Dim mNoArgs()
Dim strUrl As String
Dim lngSearchFlags As Long

lngSearchFlags = com.sun.star.frame.FrameSearchFlag.CREATE + _
   com.sun.star.frame.FrameSearchFlag.ALL
strUrl = "file:///C:/test.sxw"
objDoc = StarDesktop.loadComponentFromURL(strUrl, "MyFrame", _
   lngSearchFlags, mNoArgs)
MsgBox "Drücken Sie OK, um das zweite Dokument anzuzeigen."
strUrl = "file:///C:/test2.sxw"
objDoc = StarDesktop.loadComponentFromURL(strUrl, "MyFrame", _
   lngSearchFlags, mNoArgs)
```

Tabelle 4.38 Folgende Konstanten stehen für FrameSearchFlag zur Verfügung

Konstante	Wert	Beschreibung
PARENT	1	erlaubt die Suche im übergeordneten Rahmen
SELF	2	verwendet den selben Frame
CHILDREN	4	schließt alle untergeordneten Frames ein
CREATE	8	der Frame wird erstellt, falls er nicht gefunden wurde
SIBLINGS	16	schließt die direkten Kinder des Start-Frames ein
TASKS	32	erlaubt die Suche außerhalb der Prozedur
ALL	23	schließt alle Frames ein, außer Frames anderer Tasks; erstellt aber keinen neuen Frame
GLOBAL	55	sucht in der gesamten Hierarchie der Frames; erstellt aber keinen neuen Frame

4.2.4 loadComponentFromURL

Bei `loadComponentFromURL` handelt es sich um ein PropertyValue-Datenfeld, das OOo verschiedene Optionen für das Öffnen beziehungsweise Erzeugen von Dokumenten mit gibt. Für jede Option muss das Datenfeld eine PropertyValue-Struktur bereithalten, in der der Name der Option als Zeichenfolge sowie der zugehörige Wert abgelegt werden.

`loadComponentFromURL` unterstützt folgende Optionen:

Tabelle 4.39 Einige der Optionen (die vollständige Liste s. oben)

Option	Wert	Beschreibung
AsTemplate	Boolean	lädt eine Vorlage direkt, wenn AsTemplate auf den Wert „True“ gesetzt ist. Ansonsten bewirkt das Laden einer Vorlage das Erzeugen eines neuen Dokumentes basierend auf der Vorlage.
CharacterSet	String	legt fest, welcher Zeichensatz einem Dokument zugrunde liegt.
FilterName	String	gibt einen speziellen Filter für die Funktion loadComponentFromURL vor. Die möglichen Filternamen sind in der Datei \share\config\registry\instance\org\openoffice\office\ TypeDetection.xml definiert.
FilterOptions	String	legt Zusatzoptionen für Filter fest.
JumpMark	String	springt nach dem Öffnen eines Dokumentes zu der in JumpMark definierten Position.
Password	String	übergibt ein Passwort für das Öffnen einer geschützten Datei.
ReadOnly	Boolean	lädt ein Dokument mit Schreibschutz.

Das nächste Beispiel zeigt, wie sich eine durch Kommagetrennte Textdatei in Calc unter Verwendung der Option `FilterName` öffnen lässt.

```
Dim objDoc As Object
Dim FileProperties(0) As New com.sun.star.beans.PropertyValue
Dim strUrl As String
strUrl = "file:///C:/csv.doc"
FileProperties(0).Name = "FilterName"
FileProperties(0).Value ="scalc: Text - txt - csv (StarOffice Calc)"

objDoc = StarDesktop.loadComponentFromURL _
   (strUrl, "_blank", 0, FileProperties())
```

Ein neues leeres Dokument wird folgendermaßen erzeugt:

```
Dim mNoArgs()
Dim strUrl As String
Dim objDoc As Object
strUrl = "private:factory/swriter"
objDoc = StarDesktop.loadComponentFromURL _
   (strUrl, "_blank", 0, mNoArgs())
```

Der Zugriff auf das aktuelle Dokument, das heißt auf das „oben liegende“, erfolgt mit:

```
objDoc = StarDesktop.CurrentComponent
```

Wird kein Dokument geöffnet, sondern auf ein Dokument zugegriffen, an dem der Code hängt, dann muss mit

```
objDoc = ThisComponent
```

gearbeitet werden.

Der Dateiname kann mit der Eigenschaft `objDoc.URL` ausgelesen werden. Ein leeres Dokument, beziehungsweise eines, das auf einer Vorlage basiert, gibt eine leere Zeichenkette zurück.

Die Funktion `loadComponentFromURL` liefert ein Dokument-Objekt zurück, das den Service „com.sun.star.document.OfficeDocument“ unterstützt, der wiederum zwei zentrale Schnittstellen anbietet:

- die Schnittstelle „com.sun.star.frame.XStorable“, die für das Speichern von Dokumenten zuständig ist
- die Schnittstelle „com.sun.star.view.XPrintable“, die Methoden für das Ausdrucken von Dokumenten bereithält.

4.2.5 Dokumente speichern und exportieren

Das Speichern von OOo-Dokumenten erfolgt direkt über das Dokument-Objekt. Hierzu steht die Methode `store` der Schnittstelle „com.sun.star.frame.XStorable“ zur Verfügung:

```
objDoc.store()
```

Dieser Aufruf funktioniert nur, wenn dem Dokument bereits ein Speicherplatz zugewiesen wurde, was über die Eigenschaft URL abgefragt werden kann. Sie entspricht dem Menüpunkt DATEI | SPEICHERN.

Bei neuen Dokumenten kommt die Methode `storeAsURL` zum Einsatz, die ebenfalls in „com.sun.star.frame.XStorable“ definiert ist und über die sich der Standort des Dokumentes festlegen lässt:

```
Dim strURL As String
Dim mNoArgs()
strUrl = "file:///C:/test3.sxw"
objDoc.storeAsURL(strURL, mNoArgs())
```

Übrigens kann eine Datei mit `storeToURL` an einen anderen Ort oder unter einem anderen Namen gespeichert werden, wobei die alte Datei im Frame bleibt.

Neben der Eigenschaft `URL`, mit der überprüft werden kann, ob ein Dokument einen Dateinamen besitzt, stellt „com.sun.star.frame.XStorable“ weitere Hilfsmethoden zur Verfügung:

Methode	Beschreibung
hasLocation()	gibt an, ob dem Dokument bereits eine URL zugewiesen wurde.
getLocation()	liefert die URL.
isReadonly()	gibt an, ob ein Dokument schreibgeschützt ist.
isModified()	gibt an, ob ein Dokument nach dem letzten Speichervorgang geändert wurde.

Mit diesen Prüfmöglichkeiten lässt sich der oben stehende Code zum Speichern eines Dokumentes ausbauen, so dass die Speicherung nur dann erfolgt, wenn das Objekt tatsächlich geändert wurde, und der Dateiname nur dann abgefragt wird, wenn er auch wirklich benötigt wird:

```
If (objDoc.isModified()) Then
   If (objDoc.hasLocation() And (Not objDoc.isReadOnly())) Then
      objDoc.store()
   Else
      objDoc.storeAsURL(strURL, mNoArgs())
   End If
End If
```

Das Beispiel prüft zunächst, ob das betreffende Dokument seit dem letzten Speichervorgang geändert wurde. Nur falls dies der Fall ist, wird der Speichervorgang fortgesetzt. Wurde dem Dokument bereits eine URL zugeordnet und besitzt es keinen Schreibschutz, so wird es unter der vorhandenen URL gespeichert. Besitzt es keine URL beziehungsweise wurde es mit Schreibschutz geöffnet, so wird es unter einer neuen URL gespeichert.

4.2.6 Die Optionen der Methode storeAsURL

Wie bei der Methode `loadComponentFromURL` lassen sich auch bei der Methode `storeAsURL` (und `storeToURL`) einige Optionen in Form eines PropertyValue-Datenfeldes angeben. Diese bestimmen das Vorgehen von OOo bei der Speicherung eines Dokumentes. Für `storeAsURL` stehen unter anderem folgende Optionen zur Verfügung. Sie sind identisch mit den Optionen der Methode `loadFromURL`:

Option	Wert	Beschreibung
CharacterSet	String	legt fest, welcher Zeichensatz einem Dokument zugrunde liegt.
FilterName	String	gibt einen speziellen Filter für die Funktion loadComponentFromURL

Option	Wert	Beschreibung
		vor. Die möglichen Filternamen sind in der Datei \share\config\registry\instance\org\openoffice\office\TypeDetection.xml definiert.
FilterOptions	String	legt Zusatzoptionen für Filter fest.
Overwrite	Boolean	erlaubt das Überschreiben einer bereits existierenden Datei ohne Rückfrage.
Password	String	übergibt das Passwort für eine geschützte Datei.
Unpacked	Boolean	speichert das Dokument unkomprimiert in Unterverzeichnissen ab.

Das nächste Beispiel zeigt, wie die Option Overwrite zusammen mit storeAsURL verwendet werden kann. Das Beispiel speichert Doc auch dann unter dem angegebenen Dateinamen ab, wenn unter dem Namen bereits eine Datei existiert.

```
Dim objDoc As Object
Dim mFileProperties(0) As New com.sun.star.beans.PropertyValue
Dim strUrl As String

strUrl = "file:///c:/test3.sxw"
mFileProperties(0).Name = "Overwrite"
mFileProperties(0).Value = True
objDoc.storeAsURL(strUrl, mFileProperties())
```

Wenn Sie den Wert der Eigenschaft auf „False" setzen, dann erhalten Sie einen Hinweis, dass das Dokument bereits existiert. Anschließend folgt eine Fehlermeldung. Wird auf die Eigenschaft verzichtet, erfolgt paradoxerweise kein Hinweis!

4.3 Dokumente drucken

Ähnlich wie das Speichern erfolgt auch das Ausdrucken von Dokumenten direkt über das Dokument-Objekt. Dazu dient die Methode `print` der Schnittstelle „com.sun.star.view.XPrintable". In seiner einfachsten Form lautet der Aufruf von `print`:

```
Dim mFileProperties()
objDoc.print(mFileProperties())
```

Bei dem Parameter `mNoArgs` handelt es sich wie im Falle der Methode `loadComponentFromURL` und `storeAsURL` um ein PropertyValue-Datenfeld, über das OOo einige Optionen für das Ausdrucken spezifizieren kann.

4.3.1 Die Optionen der Methode print

Die Methode `print` erwartet ein PropertyValue-Datenfeld als Parameter, das die Einstellungen des Druck-Dialoges von OOo widerspiegelt. Hier die vollständige Liste:

Option	Wert	Beschreibung
CopyCount	Integer	gibt die Anzahl der auszudruckenden Kopien an.
FileName	String	druckt das Dokument in die genannte Datei.
Collate	Boolean	druckt doppelseitig.
Sort	Boolean	sortiert die Seiten beim Ausdruck mehrerer Exemplare (CopyCount > 1).
Pages	String	enthält die Auflistung der auszudruckenden Seiten (Syntax wie über den Druck-Dialog)

Das nächste Beispiel zeigt, wie sich über die Option `Pages` mehrere Seiten (Nummer 1 bis 3, 7 und 9) eines Dokumentes ausdrucken lassen:

```
Sub Api8
   Dim objDoc As Object
   Dim PrintProperties(2) As New com.sun.star.beans.PropertyValue
   PrintProperties(0).Name = "Pages"
   PrintProperties(0).Value = "1-3; 7; 9"
   PrintProperties(1).Name = "Sort"
   PrintProperties(1).Value = True
   PrintProperties(2).Name = "CopyCount"
   PrintProperties(2).Value = 2

   ThisComponent.print(PrintProperties())

End Sub
```

Beachten Sie, dass es auch in Calc möglich ist, ein leeres Dokument zu drucken!

4.3.2 Druckerauswahl und -einstellungen

Zur Auswahl des für einen Ausdruck zu verwendenden Druckers stellt die Schnittstelle „com.sun.star.view.XPrintable“ eine Eigenschaft Printer zur Verfügung, die ein PropertyValue-Datenfeld mit folgenden Einstellungen entgegennimmt:

Option	Wert	Beschreibung
Name	String	Name des Druckers
PaperOrientation	Enum	Seitenausrichtung (Wert com.sun.star.view.PaperOrientation.-PORTRAIT für Hochformat, com.sun.star.view.PaperOrientation.-LANDSCAPE für Querformat)
PaperFormat	Enum	Papierformat (z.B. com.sun.star.view.PaperFormat.A4 für DIN A4 oder A3, A5, B4, B5 ebenso wie com.sun.star.view.PaperFor-mat.Letter für US-Letter)
PaperSize	Size	Papiergröße in hundertstel Millimeter: Width und Height

Das nächste Beispiel zeigt, wie sich mit Hilfe der Printer-Eigenschaft ein Drucker wechseln und die Papiergröße einstellen lässt.

```
Dim objDoc As Object
Dim PrinterProperties(1) As New com.sun.star.beans.PropertyValue
Dim PaperSize As New com.sun.star.awt.Size

PaperSize.Width = 20000  ' entspricht 20 cm
PaperSize.Height = 20000 ' entspricht 20 cm
PrinterProperties (0).Name="Name"
PrinterProperties (0).Value="Epson Sylus COLOR 600 ESC/P 2"
PrinterProperties (1).Name="PaperSize"
PrinterProperties (1).Value=PaperSize
objDoc.Printer = PrinterProperties()
```

Das Beispiel definiert ein Objekt namens `PaperSize` mit dem Typ „com.sun.star.awt.Size“. Dies wird zur Angabe der Papiergröße benötigt. Des Weiteren erstellt es ein Datenfeld für zwei PropertyValue-Einträge namens `PrinterProperties`. Dieses Datenfeld wird anschließend mit den zu setzenden Werten initialisiert und der Eigenschaft Printer zugewiesen (aus UNO-Sicht ist Printer keine echte, sondern eine nachgebildete Eigenschaft).

4.3.3 Druckbereich

Die Schnittstelle „com.sun.star.sheet.XPrintAreas“ ist für Calc-Tabellenblätter verfügbar. Sie stellt den Zugriff auf die druckbaren Zellen zur Verfügung, die zur Struktur „com.sun.star.table.CellRangeAddress“ gehören. Dazu zählen:

Tabelle 4.40 Methoden der Struktur „com.sun.star.table.CellRangeAddress“:

Methode	Beschreibung
getPrintAreas()	gibt die Druckbereiche des Tabellenblattes zurück
setPrintAreas()	setzt die Druckbereiche
getPrintTitleColumns() getPrintTitleRows()	gibt „True“ zurück, wenn Spalten oder Zeilen wiederholt werden
setPrintTitleColumns() setPrintTitleRows()	schaltet Wiederholungsspalten oder -zeilen ein
getTitleColumns() getTitleRows()	gibt die Wiederholungsspalten oder -zeilen zurück
setTitleColumns() setTitleRows()	schaltet die markierten Spalten oder Zeilen als Wiederholungsspalten ein.

Weitere Informationen zum Drucken von Tabellenblättern finden Sie in Kapitel 5.

4.4 Schließen

Für das Schließen des Dokuments ist die Methode `Close` zuständig – sie verlangt einen Parameter vom Typ „Boolean“:

```
Sub Api9
   ThisComponent.Close(True)
End Sub
```

Selbstverständlich sollten Sie das Dokument vor dem Schließen speichern, da OOo nicht nachfragt, ob Sie die Änderungen speichern oder verwerfen möchten. Sollte allerdings noch ein Prozess innerhalb des Dokumentes laufen, gibt es keine Möglichkeit, diesen zu stoppen. Dann führt die Methode `Close` allerdings zu einer `CloseVetoException`. Sollte ein solcher Fehler auftreten, dann bedeuten die Parameter „True“ oder „False“ Folgendes:

„True“ verschiebt den Zeiger des zu schließenden Objektes auf dasjenige, das die `CloseVetoException` generiert. Dieser neue Eigentümer muss nun das Objekt erneut schließen, damit der laufende Prozess beendet wird.

„False“ belässt den Eigentümer des zu schließenden Objekts. Allerdings muss man nun auf mögliche `CloseVetoExceptions` reagiert werden und – möglicherweise – das Objekt zu einem späteren Zeitpunkt schließen.

Falls in OOo nur dieses eine Dokument offen war, dann bewirkt die Methode Close, dass die gesamte Applikation geschlossen wird – anders als in OOo selbst, wo das Schließen des letzten Dokuments noch den StarDeskop offen lässt.

4.5 OOo beenden

Sie müssen erst alle laufenden Prozesse beenden, das heißt, die Methode `Close` muss auf alle offenen Objekte angewendet werden. Dann dürfen Sie OOo mit der Methode `dispose()` beenden:

```
Sub Api10

   ThisComponent.Close(True)

   StarDesktop.dispose()

End Sub
```

Diese Methode macht nur dann wirklich Sinn, wenn Sie „von außen“ OOo schließen – innerhalb einer Anwendung werden Sie wohl kaum dem Anwender „sein“ OOo beenden!

Erzeugen kontextunabhängiger Objekte

Während Texttabellen in Textdateien Einzug finden und dort in erster Linie zum Halten und Darstellen von Daten innerhalb eines Textdokuments dienen, während Datenbanktabellen Datenbankfunktionalität anbieten, so liegt die Stärke von Calc-Tabellen sicherlich in der Darstellung und Berechnung von „kleineren“ Datenmengen, das heißt von einigen Dutzend bis mehreren tausend Daten. Dem Zugriff auf diese Tabellen ist das folgende Kapitel gewidmet.

5 Calc

Das oberste Objekt ist das Dokument. Seine wichtigsten Methoden (Öffnen, Speichern, Drucken, Schließen) wurde bereits in Kapitel 4 beschrieben.

Wenn Sie überprüfen möchten, ob es sich bei einem Objekt wirklich um ein „Calc Dokument“ handelt, dann finden Sie es heraus, indem Sie überprüfen, ob es den Service „com.sun.star.sheet.SpreadsheetDocument“ unterstützt:

```
If Thiscomponent.SupportsService _
   ("com.sun.star.sheet.SpreadsheetDocument") = False Then
   MsgBox "Sie sind nicht im richtigen Dokument"
Else
...
```

5.1 Der Zugriff auf die Tabellen

Auf die Liste der Tabellenblätter wird über die Sammlung `Sheets` oder die Methode `getSheets()` zugegriffen. Jedes einzelne Blatt kann mit seiner Nummer (`getByIndex` – die Zählung beginnt bei 0!) oder mit seinem Namen (`getByName`) angesprochen werden:

```
Sub Blatt_Zugriff1

   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object

   objDatei = ThisComponent
   objBlaetter = objDatei.Sheets
   objBlatt = objBlaetter.getByName("Tabelle1")

End Sub
```

oder so:

```
   objBlatt = objDatei.Sheets("Tabelle1")
```

oder so:

```
   objBlaetter = objDatei.getSheets()
```

Es funktioniert auch mit der Methode `getByName`:

```
objBlatt = objDatei.Sheets.getByName("Tabelle1")
```

Der Name wird mit der Eigenschaft `Name` ermittelt:

```
   objBlatt = objDatei.Sheets(0)
   MsgBox objBlatt.Name
```

oder über den Index:

```
objBlatt = objDatei.Sheets.getByIndex(0)
```

Das über `getByName` oder `getByIndex` gewonnene Sheet-Objekt unterstützt den Service „com.sun.star.sheet.Spreadsheet“.

Tabelle 5.41 Das Sheet-Objekt stellt folgende wichtige Eigenschaften zur Verfügung:

Eigenschaft	**Beschreibung**
IsVisible (Boolean)	das Spreadsheet ist sichtbar oder unsichtbar
PageStyle (String)	Name der Seitenvorlage des Spreadsheets

Einen Überblick sämtlicher Namen der Tabellen der Datei liefert die Methode `getElementNames`. Als Ergebnis liefert sie ein Datenfeld mit den Namen. Mit Hilfe dieses Datenfeldes können alle Blattnamen eines Tabellendokuments ausgelesen werden:

```
Sub Blatt_Zugriff02

   Dim objBlaetter As Object
   Dim strBlattNamen() As String
   Dim strBlattListe As String
   Dim i As Integer

   objBlaetter = ThisComponent.Sheets
   strBlattNamen = objBlaetter.getElementNames
   For i = LBound(strBlattNamen) To UBound(strBlattNamen)
      strBlattListe = strBlattListe & Chr(13) & strBlattNamen(i)
   Next i
```

```
   MsgBox strBlattListe

End Sub
```

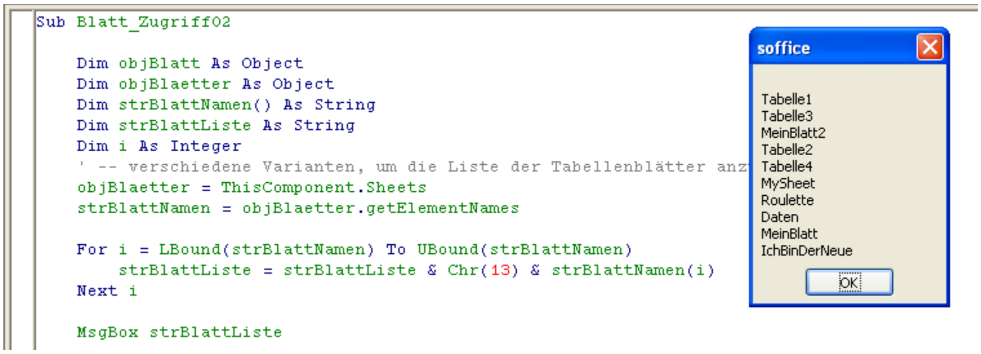


Abbildung 5.68 Die Liste der Tabellenblätter

Der Zugriff funktioniert auch mit `getByIndex`:

```
   For i = 0 to objBlaetter.getCount() - 1
      objBlatt = objBlaetter.getByIndex(i)
      strBlattListe =  strBlattListe & Chr(13) & objBlatt.Name
   Next i

   MsgBox strBlattListe
```

Mit der Methode `CreateEnumeration` wird ein Aufzählungsobjekt erstellt, das mit der Methode `nextElement()` von vorne nach hinten durchlaufen werden kann. Jedes Objekt kann an eine Objektvariable übergeben und weiter verarbeitet werden. Um zu überprüfen, ob noch weitere Objekte vorhanden sind, steht die Methode `hasMoreElements()` zur Verfügung:

```
   objBlaetter = ThisComponent.Sheets.CreateEnumeration
   Do
      objBlatt = objBlaetter.nextElement
      strBlattListe =  strBlattListe & Chr(13) & objBlatt.Name
   Loop While objBlaetter.hasMoreElements()
```

```
   MsgBox strBlattListe
```

Die Methode `hasByName(String)` verrät, ob ein Tabellenblatt mit einem bestimmten Namen innerhalb der Datei vorhanden ist. So gibt das folgende Beispiel eine Meldung darüber aus, ob die Sammlung „objBlaetter“ ein Tabellenblatt mit dem Namen „Tabelle99“ enthält.

```
Sub Blatt_Zugriff03
   Dim objBlaetter As Object

   objBlaetter = ThisComponent.Sheets

   If objBlaetter.HasByName("Tabelle99") Then
      MsgBox "Das Blatt ""Tabelle99"" ist vorhanden."
   Else
      MsgBox "Das Blatt ""Tabelle99"" ist nicht vorhanden."
   End If
End Sub
```

5.1.1 Die Anzahl der Tabellenblätter

Die Eigenschaft `Count` und die Methode `getCount()` ermitteln die Anzahl der Tabellenblätter:

```
   MsgBox objBlaetter.Count
   MsgBox objBlaetter.getCount()
```

Beachten Sie, dass das erste Blatt die Indexnummer 0 besitzt, so dass die Zählung nur bis `getCount() - 1` beziehungsweise `Count - 1` hochzählen darf.

5.1.2 Blätter umbenennen

Um ein Blatt umzubenennen, muss lediglich die Eigenschaft `Name` gesetzt oder mit der Methode `setName(String)` der neue Name übergeben werden:

```
objBlatt.Name = "Hugo"
```

Natürlich sollte vorher mit der Methode `HasByName` überprüft werden, ob bereits ein Blatt mit einem solchen Namen existiert.

5.1.3 Blätter löschen, verschieben und kopieren

Das Löschen eines Blattes erfolgt mit der Methode `removeByName`. Falls es nicht existiert, erfolgt ein Fehler.

```
Sub Blatt_Zugriff04
   Dim objBlaetter As Object
   objBlaetter = ThisComponent.Sheets
   If objBlaetter.HasByName("Tabelle2") Then
      objBlaetter.removeByName("Tabelle2")
      MsgBox "Das Blatt wurde erfolgreich gelöscht."
   End If
End Sub
```

Während Calc immer beim Löschen nachfragt, ob das Blatt wirklich gelöscht werden soll (auch wenn es keine Informationen enthält), wird in Basic bei der Methode `removeByName` nie nachgefragt.

Ein Blatt wird mit der Methode `insertByName` eingefügt:

```
Sub Blatt_Zugriff05
   Dim objDatei As Object
   Dim objBlatt As Object

   objDatei = ThisComponent
   objBlatt = ThisComponent.Sheets(0)

   If objDatei.Sheets.hasByName("MeinBlatt") Then
      objBlatt = objDatei.Sheets.getByName("MeinBlatt")
   Else
      objBlatt = objDatei.createInstance _
         ("com.sun.star.sheet.Spreadsheet")
      objDatei.Sheets.insertByName("MeinBlatt", objBlatt)
   End If
End Sub
```

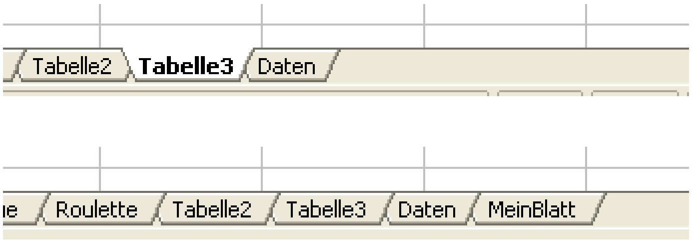


Abbildung 5.69 Ein neues Blatt wird eingefügt.

Etwas einfacher kann ein neues Blatt mit der Methode `insertNewByName(String, Index)` eingefügt werden. Sie verlangt lediglich den neuen Namen und die Position (die Zählung beginnt wieder bei 0!):

```
objDatei.Sheets.insertNewByName("MeinBlatt2", 1)
```

Ein Blatt wird durch ein leeres Blatt mit der Methode `replaceByName` ersetzt (und damit werden sämtliche Inhalte ohne Nachfragen gelöscht):

```
Sub Blatt_Zugriff06
   Dim objDatei As Object
   Dim objBlatt As Object

   objDatei = ThisComponent
   If objDatei.Sheets.hasByName("Tabelle3") Then
      objBlatt = _
         objDatei.createInstance("com.sun.star.sheet.Spreadsheet")
      objDatei.Sheets.replaceByName("Tabelle3", objBlatt)

   End If
End Sub
```

Das nächste Beispiel erzeugt ein neues Tabellendokument. Alle Blätter bis auf eines werden gelöscht, neun neue Blätter werden erzeugt – die Blätter werden umbenannt in „Blatt01", Blatt02" … „Blatt10":

```
Sub Blatt_Zugriff07
   Dim objDesktop As Object
   Dim objDatei As Object
   Dim objBlaetter As Object
```

```
   Dim objBlatt As Object
   Dim strURL As String
   Dim mNoArgs()
   Dim i As Integer

   objDesktop = createUnoService("com.sun.star.frame.Desktop")
   strURL = "private:factory/scalc"
   objDatei = objDesktop.loadComponentFromURL _
      (strURL, "_blank", 0, mNoArgs())
   objBlaetter = objDatei.Sheets

   For i = objBlaetter.getCount() - 1 To 1 Step -1
      objBlaetter.removeByName(objDatei.Sheets(i).Name)
   Next i
   objDatei.Sheets(i).Name = "Blatt01"
   For i = 2 To 10
      objBlatt = _
         objDatei.createInstance("com.sun.star.sheet.Spreadsheet")
      objDatei.Sheets.insertByName("Blatt" & Format(i, "00"), objBlatt)
   Next i
End Sub
```

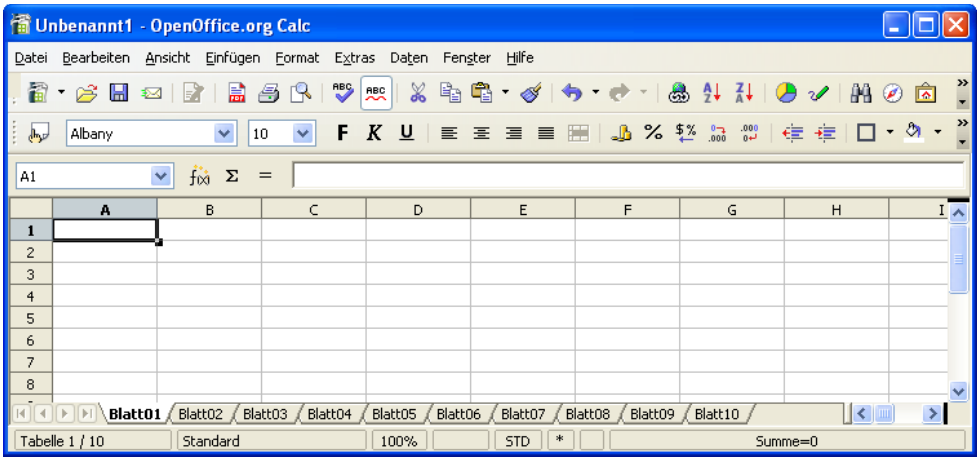


Abbildung 5.70 Eine neue Datei mit zehn neuen, leeren Blättern wird erzeugt.

Wenn Sie versuchen, alle Blätter eines Calc-Dokuments zu löschen, dann werden Sie beim letzten Blatt eine Fehlermeldung erhalten. Ebenso, wie Sie nicht innerhalb von Calc das letzte Blatt löschen können, verlangt das Dokument immer mindestens ein Tabellenblatt. Deshalb ist die Methode `hasElements()` überflüssig, mit der überprüft werden könnte, ob noch ein Tabellenblatt existiert:

```
objBlaetter.hasElements()
```

Sie liefert immer „True“.

Mit der Methode `movebyName` kann ein Blatt an eine andere Stelle verschoben werden. Die Position wird als Integer-Wert angegeben. Im folgenden Beispiel wird „Tabelle1“ – falls sie existiert – an die letzte Position verschoben:

```
Sub Blatt_Zugriff08

   Dim objDatei As Object
   Dim objBlatt As Object

   objDatei = ThisComponent

   If objDatei.Sheets.hasByName("Tabelle1") Then
      objDatei.Sheets.moveByName("Tabelle1", objDatei.Sheets.Count)
   End If
```

```
End Sub
```

Analog funktioniert `copyByName`:

```
Sub Blatt_Zugriff09

   Dim objDatei As Object
   Dim objBlaetter As Object

   objDatei = ThisComponent
   objBlaetter = objDatei.Sheets

   If objBlaetter.hasByName("Tabelle1") And _
      objBlaetter.hasByName("IchBinDerNeue") = False Then
      objBlaetter.copyByName("Tabelle1", "IchBinDerNeue", _
         objBlaetter.Count)
   End If

End Sub
```

Beachten Sie, dass zuerst die Position ermittelt wird, an die das vorhandene Blatt geschoben oder kopiert wird, dann erst wird es verschoben und kopiert. Denn nun ist die Nummerierung der Blätter eine andere!

Übrigens scheint es nicht möglich zu sein, ein Tabellenblatt von einer Datei in eine andere zu schieben.

Ein Befehl

```
   objDatei.Sheets.insertByName("DerNeue", objBlatt)
```

führt zu einer Fehlermeldung, wenn `objBlatt` auf ein Blatt einer anderen Datei verweist.

5.1.4 Sichtbare und unsichtbare Blätter

Ein Blatt kann mit der Eigenschaft `IsVisible` ausgeblendet werden:

```
Sub Blatt_Zugriff10
   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object

   objDatei = ThisComponent
   objBlaetter = objDatei.Sheets
   objBlatt = objDatei.Sheets(0)

   If objBlaetter.getCount() > 1 Then
      objBlatt.IsVisible = False
   End If

End Sub
```

Übrigens hat der Benutzer nun keine Möglichkeit mehr, dieses Blatt in Calc wieder einzublenden.

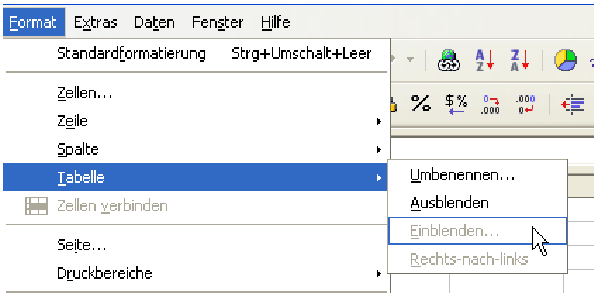


Abbildung 5.71 Obwohl ein Blatt ausgeblendet wurde, kann es der Benutzer nicht wieder einblenden.

Umgekehrt können mit einem Makro schnell sämtliche ausgeblendete Tabellenblätter wieder eingeblendet werden:

```
Sub Blatt_Zugriff11
```

```
   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim i As Integer

   objDatei = ThisComponent
   objBlaetter = objDatei.Sheets

   For i = 0 To objBlaetter.getCount() -1
      If objDatei.Sheets(i).IsVisible = False Then
         objDatei.Sheets(i).IsVisible = True
      End If
   Next

End Sub
```

Die Überprüfung

```
      If objDatei.Sheets(i).IsVisible = False Then
```

ist dabei überflüssig – es geht auch direkt:

```
   For i = 0 To objBlaetter.getCount() -1
      objDatei.Sheets(i).IsVisible = True
   Next
```

Es ist wichtig zu wissen, dass in ein verborgenes Blatt Werte geschrieben und aus ihm Daten ausgelesen werden können:

```
Sub Blatt_Zugriff13

   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object
   Dim objZelle As Object
```

```
   objDatei = ThisComponent
   objBlaetter = objDatei.Sheets
   objBlatt = objDatei.Sheets(0)

   If objBlaetter.getCount() > 1 Then
      objBlatt.IsVisible = False
      objZelle = objBlatt.getCellByPosition(0, 0)
      objZelle.String = "Ich bin der verborgene Text."
      MsgBox objZelle.String
   End If

End Sub
```

Beachten Sie, dass sich durch das Verbergen eines Blattes nichts an der Reihenfolge der Blätter ändert – wird das erste Blatt („Sheets(0)“) ausgeblendet, dann bleibt es das erste Blatt. Auch die Anzahl der Blätter verändert sich dadurch natürlich nicht! Das bedeutet, dass Sie bei komplexen Aktionen überprüfen sollten, ob möglicherweise das Blatt, auf das Sie zugreifen, ausgeblendet ist, da sonst Aktionen wie verschieben, kopieren, ausblenden und so weiter obsolet sind.

5.1.5 Blattschutz

Der Schutz wird mit der Methode `protect` eingeschaltet, mit `unprotect` wird er aufgehoben. Die Methode `isProtected()` überprüft, ob ein Schutz auf dem Tabellenblatt liegt:

```
Sub Blatt_Zugriff12
   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object
   objDatei = ThisComponent
   objBlaetter = objDatei.Sheets
   objBlatt = objDatei.Sheets(0)

   If objBlatt.isProtected() = True Then
      objBlatt.unprotect("")
      MsgBox "Schutz auf"
```

```
   Else
      objBlatt.protect("")
      MsgBox "Schutz zu"
   End If

End Sub
```

Beachten Sie, dass die Methoden `protect()` und `unprotect()` immer ein Kennwort verlangen. Wird kein Kennwort festgelegt, dann wird eine leere Zeichenkette übergeben. Ein Schutz mit Kennwort sieht folgendermaßen aus:

```
   If objBlatt.isProtected() = True Then
      objBlatt.unprotect("Passwort")
      MsgBox "Schutz auf"
   Else
      objBlatt.protect("Passwort")
      MsgBox "Schutz zu"
   End If
```

Übrigens: Wird ein Tabellenblatt, auf dem ein Schutz mit Kennwort liegt, mit einem falschen Kennwort geöffnet, so geht das Programm ohne Fehlermeldung darüber hinweg. Jedoch kann leicht überprüft werden, ob das Aufheben des Kennwortschutzes erfolgreich war:

```
      objBlatt.unprotect("Password")
      MsgBox objBlatt.isProtected()
```

Wird umgekehrt ein Blatt, auf dem ein Schutz ohne Kennwort liegt, mit der Methode `unprotect` und einem Kennwort aufgehoben, so funktioniert dies problemlos.

5.2 Zeilen und Spalten in Tabellenblättern

Jedes Tabellenblatt enthält eine Liste mit seinen Zeilen und Spalten. Sie sind über die Sammlungen „Rows" beziehungsweise „Columns" des Spreadsheet-Objektes verfügbar und unterstützen die Services „com.sun.star.table.TableRows" beziehungsweise „com.sun.star.table.TableColumns".

5.2.1 Zugriff auf Spalten und Zeilen

Im folgenden Beispiel verweisen die beiden Objektvariablen `objErsteZeile` und `objErsteSpalte` auf die erste Zeile und die erste Spalte eines Tabellenblatts:

```
Sub ZeilenUSpalten01

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objErsteZeile As Object
   Dim objErsteSpalte As Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)

   objErsteSpalte = objBlatt.Columns(0)
   objErsteZeile = objBlatt.Rows(0)
End Sub
```

Die Spaltenobjekte unterstützen den Service „com.sun.star.table.TableColumn“.

Tabelle 5.42 Die Spalten halten folgende Eigenschaften bereit:

Eigenschaft	Wert	Beschreibung
Width	Long	Breite der Spalte in 100stel Millimeter
OptimalWidth	Boolean	setzt die Spalte auf ihre optimale Breite
IsVisible	Boolean	zeigt die Spalte an oder blendet sie aus
IsStartOfNewPage	Boolean	erzeugt beim Ausdrucken vor der Spalte einen Seitenumbruch

Die Zeilenobjekte basieren auf dem Service „com.sun.star.table.RowColumn“.

Tabelle 5.43 Die Zeilen verfügen über die Eigenschaften:

Eigenschaft	Wert	Beschreibung
Height	Long	Höhe der Zeile in 100stel Millimeter
OptimalHeight	Boolean	setzt die Zeile auf ihre optimale Höhe
IsVisible	Boolean	zeigt die Zeile oder blendet sie aus
IsStartOfNewPage	Boolean	erzeugt beim Ausdrucken vor der Zeile einen Seitenumbruch

Die Methode `getName()` kann verwendet werden, um den Namen der Spalte zu ermitteln. Dabei liefert:

```
objErsteSpalte = objBlatt.Columns(0)
```

den Buchstaben „A“ und

```
objErsteSpalte = objBlatt.Columns(26)
```

die Buchstabenfolge „AA“.

Beachten Sie, dass die Methode `getName()` für Zeilen nicht existiert!

5.2.2 Spaltenbreite und Zeilenhöhe

Die Funktionen zur automatischen Anpassung der Spaltenbreite unterscheidet sich leicht von der der Zeilenhöhe. Die Spaltenbreite wird beim Setzen der Eigenschaft `OptimalWidth` auf „True“ auf den längsten Eintrag innerhalb der Spalte angepasst. Verändern sich Inhalte einzelner Zellen, so hat dies keine Auswirkungen auf die Spaltenbreite.

Erhält die Eigenschaft `OptimalHeight` einer Zeile den Wert „True“, so ändert sich die Zeilenhöhe automatisch, sofern eine Zelle anders formatiert oder der Inhalt geändert wird. Allerdings wird eine Mindesthöhe von 0,45 cm nicht unterschritten. Die automatische Anpassung bleibt so lange gültig, bis die Zeile über die Eigenschaft `Height` eine absolute Höhe erhält.

Das folgende Beispiel demonstriert den Umgang mit den Eigenschaften `OptimalWidth` und `OptimalHeight`. Es schaltet die automatische Breite für die ersten fünf Spalten ein, vergrößert die sechste Spalte (F) auf 8 cm, blendet die zweite Zeile aus und vergrößert die vierte Zeile auf 2 cm.

```
Sub ZeilenUSpalten02
   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objZeile As Object
   Dim objSpalte As Object
   Dim i As Integer

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)

   For i = 0 To 4
```

```
      objSpalte = objBlatt.Columns(i)
      objSpalte.OptimalWidth = True
   Next i

   objSpalte = objBlatt.Columns(5)
   objSpalte.Width = 8000

   objZeile = objBlatt.Rows(1)
   objZeile.IsVisible = False
   objZeile = objBlatt.Rows(3)
   objZeile.Height = 2000
End Sub
```

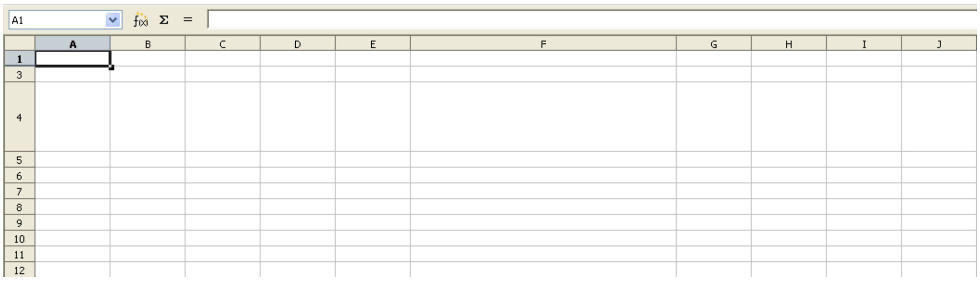


Abbildung 5.72 Unterschiedliche Zeilenhöhen und Spaltenbreiten

Analog können Sie auch die Eigenschaften wie folgt festlegen:

```
objBlatt.Columns(5).setPropertyValue("Width", 6000)
objBlatt.Columns(1).setPropertyValue("IsVisible", False)
```

5.2.3 Zeilen und Spalten einfügen und löschen

Die Objekte „Rows“ und „Columns“ des Tabellenblatts unterstützen nicht nur den Zugriff auf existierende Zeilen beziehungsweise Spalten. Sie gestatten auch das Einfügen und Löschen von neuen Spalten und Zeilen.

```
Sub ZeilenUSpalten03
   Dim objDatei as Object
```

```
   Dim objBlatt As Object
   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)

   objBlatt.Columns.insertByIndex(3, 1)
   objBlatt.Columns.removeByIndex(5, 1)

End Sub
```

Das Beispiel fügt über die Methode `insertByIndex` eine neue Spalte ein, die an vierter Position in der Tabelle erscheint (Index 3, da die Nummerierung mit 0 beginnt). Der zweite Parameter von `insertByIndex` gibt die Anzahl der einzufügenden Spalten an (in diesem Beispiel eine).

Der Aufruf `removeByIndex` löscht die sechste Spalte (Index 5, da die Nummerierung mit 0 beginnt). Auch hier gibt der zweite Parameter die Anzahl der betroffenen Spalten an.

Die Zählung beginnt (nach dem Einfügen und Löschen von Zeilen und Spalten) neu – es wird von dem veränderten Blatt „objBlatt" ausgegangen – nicht von dem ursprünglichen!

Analog arbeiten die Methoden zum Einfügen und Löschen von Zeilen über das Rows-Objekt.

5.3 Zellen

Ein Tabellenblatt setzt sich zusammen aus einer zweidimensionalen Liste mit Zellen. Jede Zelle ist über ihre X- und Y-Position eindeutig bestimmt. Die linke obere Zelle hat die Position (0 / 0) oder den Namen A1. Das nächste Beispiel erzeugt ein Objekt, das auf die linke obere Zelle („A1") verweist und dort einen Text einfügt:

```
Sub Zellzugriff01
   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object
   Dim objZelle As Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
```

```
    objZelle = objBlatt.getCellByPosition(0, 0)
    objZelle.String = "Hallo Leute, ich bin der Neue!"
    objBlatt.Columns(0).OptimalWidth = True
End Sub
```

Ebenso wie der Name der Zelle A1 lautet, wird sie über `getCellByPosition(0, 0)` zuerst mit dem Wert für die Spalte und anschließend mit dem Zeilenwert angesprochen. Wie in der deutschen Darstellung steht in der ersten Position die Spalte und in der zweiten die Zeile – A1 und nicht R1C1.

Die Zelle kann ebenso über ihren Namen angesprochen werden:

```
    objZelle = objBlatt.getCellRangeByName("B1")
    objZelle.String = "Hallo Leute, ich bin auch eine Überschrift!"

    objBlatt.Columns(1).OptimalWidth = True
```

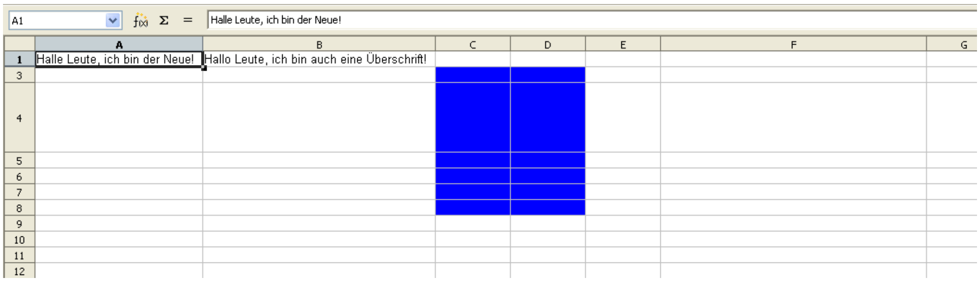


Abbildung 5.73 Text wird eingefügt und die Spalte auf optimale Breite gesetzt.

Wenn Sie über den Namen auf eine Zelle zugreifen möchten, dann entweder relativ oder absolut, das heißt mit „B1“ oder „$B$1“.

Calc unterscheidet nicht zwischen einer Zelle und einem Zellbereich. Das heißt, Sie können mit

```
    objZelle = objBlatt.getCellRangeByName("B1:C7")
    objZelle = objBlatt.getCellRangeByName("$B$1:$C$7")
```

auf einen Bereich zugreifen. Wenn Sie dies per Zahlen erledigen möchten, dann so:

```
    objZelle = objBlatt.getCellRangebyPosition(2, 1, 3, 7)
```

Allerdings kann dann nicht mehr in alle Zellen gleichzeitig geschrieben werden! Viele andere Eigenschaften (beispielsweise Formatierungen) können jedoch gesetzt werden. Im folgenden Beispiel wird der Bereich blau formatiert:

```
objZelle = objBlatt.getCellRangebyPosition(2, 1, 3, 7)

objZelle.CellBackColor = RGB(0, 0, 255)
```

Umgekehrt kann die Zellposition mit der Struktur `getRangeAddress()` ermittelt werden:

```
Sub WerBinIch
   Dim objDatei As Object
   Dim objBlatt As Object
   Dim objBereich as Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets.getByIndex(0)
   objBereich = objBlatt.getCellRangebyName("A1:Z99")

   MsgBox "Der Bereich beginnt bei Zeile: " & _
      objBereich.getRangeAddress().StartRow & _
      ", endet bei Zeile: " & _
      objBereich.getRangeAddress().EndRow & "." & _
      Chr(13) & _
      "Der Bereich beginnt bei Spalte: " & _
      objBereich.getRangeAddress().StartColumn & _
      ", endet bei Spalte: " & _
      objBereich.getRangeAddress().EndColumn & "." & _
      Chr(13) & _
      "Der Bereich liegt auf Blatt: " & _
      objBereich.getRangeAddress().Sheet & "."

End Sub
```

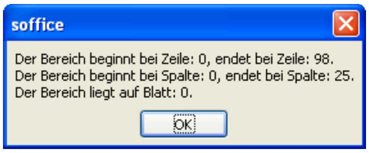


Abbildung 5.74 Die Informationen des Bereichs werden bestimmt.

Tabelle 5.44 Elemente von getRangeAddress

Element	Beschreibung
StartRow	obere Zeile
EndRow	untere Zeile
StartColumn	linke Spalte
EndColumn	rechte Spalte
Sheet	zugehöriges Blatt

Bei einer Zelle können Sie ebenfalls die fünf oben stehenden Methoden von `getRangeAddress` verwenden oder die Methoden `getCellAddress().Column`, `getCellAddress().Row`, `getCellAddress().Sheet`.

Beachten Sie, dass im Englischen „Address“ mit zwei „d“ geschrieben wird – alle Methoden und Eigenschaften enthalten folglich „address“ und nicht „adress“!

Vergessen Sie nicht, dass die Zählung der Spalten und Zeilen immer bei 0 beginnt; die Zelle C5 liegt also in Spalte 2 und in Zeile 4!

5.3.1 Zellen füllen und auslesen

Eine Tabellenzelle kann in Calc Text, Zahlen, Formeln oder gar nichts enthalten. Je nachdem, was in die Zelle geschrieben wird, muss eine andere Eigenschaft verwendet werden: Text wird über die Eigenschaft `String` eingetragen, Zahlen als `Value` und Formeln über `Formula`:

```
Sub Zellzugriff02

   Dim objDatei as Object
   Dim objBlaetter As Object
```

```
    Dim objBlatt As Object
    Dim objZelle As Object

    objDatei = ThisComponent
    objBlatt = objDatei.Sheets(0)

    objZelle = objBlatt.getCellByPosition(0, 0)
    objZelle.String = "Ausgaben Januar Netto"
    objZelle = objBlatt.getCellByPosition(0, 1)
    objZelle.Value = 25896
    objZelle = objBlatt.getCellByPosition(0, 2)
    objZelle.Value = 79825.45
    objZelle = objBlatt.getCellByPosition(0, 3)
    objZelle.Value = -6358.25
    objZelle = objBlatt.getCellByPosition(0, 4)
    objZelle.Formula = "=SUM(A2:A4)"
    objZelle = objBlatt.getCellByPosition(0, 5)
    objZelle.String = "MWSt:"

    objZelle = objBlatt.getCellByPosition(0, 6)
    objZelle.Formula = "=A5*16%"

    objBlatt.Columns(0).OptimalWidth = True

End Sub
```

	A	B
1	Ausgaben Januar Netto	
2	25896	
3	79825,45	
4	-6358,25	
5	99363,2	
6	MWSt:	
7	15898,11	
8		

Abbildung 5.75 Texte, Daten und Formeln werden eingefügt.

Das Beispiel fügt in A1 und A6 einen Text zur Beschriftung ein, in A2, A3 und A4 werden Werte eingetragen und in A5 und A7 jeweils eine Formel.

Die Eigenschaften `Value`, `String` und `Formula` ersetzen die Methode `PutCell()` zum Setzen der Werte einer Tabellenzelle.

Calc betrachtet den über die Eigenschaft `String` eingetragenen Inhalt einer Zelle als Text; das gilt auch dann, wenn es sich um eine Zahl handelt. Dies sieht man auch daran, dass die Ausgabe einer solchen Zahl linksbündig und nicht – wie bei „echten“ Zahlen üblich – rechtsbündig erfolgt. Aber auch beim Einsatz von Formeln ist auf den Unterschied zwischen Text und Zahlen zu achten:

```
Sub Zellzugriff03

   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object
   Dim objZelle As Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)

   objZelle = objBlatt.getCellRangeByName("A1")
   objZelle.String = "Ausgaben Januar Netto"
   objZelle = objBlatt.getCellByPosition(0, 1)
   objZelle.String = 25896
   objZelle = objBlatt.getCellByPosition(0, 2)
   objZelle.String = 79825.45
   objZelle = objBlatt.getCellByPosition(0, 3)
   objZelle.String = -6358.25
   objZelle = objBlatt.getCellByPosition(0, 4)
   objZelle.Formula = "=SUM(A2:A4)"
   objZelle = objBlatt.getCellByPosition(0, 5)
   objZelle.String = "MWSt:"
```

```
   objZelle = objBlatt.getCellByPosition(0, 6)
   objZelle.Formula = "=A5*16%"

   objBlatt.Columns(0).OptimalWidth = True

End Sub
```

	A	B
1	Ausgaben Januar Netto	
2	25896	
3	79825,45	
4	-6358,25	
5	0	
6	MWSt:	
7	0	
8		

Abbildung 5.76 Zahlen werden als Text eingefügt – folglich kann mit ihnen nicht gerechnet werden.

Die beiden Formeln ergeben 0, da in den drei Zellen, auf die sie zugreifen, keine Zahlen, sondern Texte stehen.

Wenn unklar ist, ob eine Zelle eine Zahl oder eine Zeichenfolge enthält, so hilft die Eigenschaft `Type` oder die Methode `getType()` weiter:

```
Function Zellentest(ByVal objZelle As Object) As String
   Dim strTyp As String

   Select Case objZelle.Type
   Case com.sun.star.table.CellContentType.EMPTY
      strTyp = "leer"
   Case com.sun.star.table.CellContentType.VALUE
      strTyp = "Zahl: " & objZelle.Value
   Case com.sun.star.table.CellContentType.TEXT
      strTyp = "Text: " & objZelle.String
   Case com.sun.star.table.CellContentType.FORMULA
      strTyp = "Formel: " & objZelle.Formula & " Ergebnis: " & _
         objZelle.Value
   End Select
```

```
   Zellentest = strTyp

End Function



Sub Zellzugriff04

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objZelle As Object
   Dim i As Integer
   Dim strAusgabe As String

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)

   For i = 0 To 10
      objZelle = objBlatt.getCellByPosition(0, i)
      strAusgabe = strAusgabe & Chr(13) & _
         "A" & i & ": " & Zellentest(objZelle)
   Next i

   MsgBox strAusgabe
End Sub
```

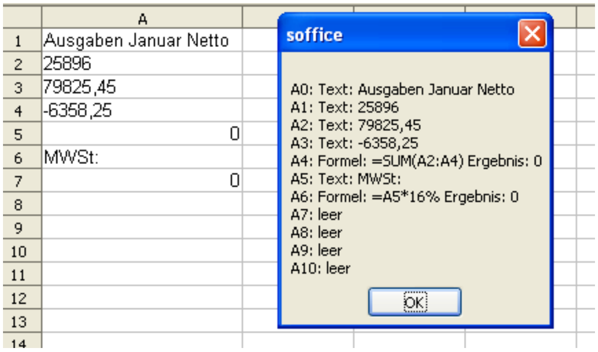


Abbildung 5.77 Text, Zahl oder Formel?

`Cell.Type` liefert einen Wert der Aufzählung „com.sun.star.table.CellContentType", der verrät, um was für einen Wert es sich handelt.

Tabelle 5.45 Die möglichen Werte von Cell.Type lauten:

Konstante	**Wert**	**Bedeutung**
EMPTY	0	kein Wert
VALUE	1	Zahl (auch Datums- und Uhrzeitangaben oder Zahlen der Form 2,14 E-007), allerdings keine komplexen Zahlen der Form 4+2i
TEXT	2	Zeichenfolge (auch als Text eingegebene oder formatierte Zahlen)
FORMULA	3	Formel (auch fehlerhafte Formel)

Tabelle 5.46 Folgende Ergebnisse werden für eine Zelle ermittelt:

Zellinhalt	**getString**	**getValue**	**getFormula**
leer	leer	0	leer
7	7	7	7
'7	7	0	'7
sieben	sieben	0	sieben
=A7 (Ergebnis: 7)	7	7	=A7
=sieben	#Name?	0	=sieben

Wenn Ihnen die Eigenschaft `Formula` zu undurchsichtig ist, dann können Sie stattdessen mit der Eigenschaft `FormulaLocal` arbeiten:

```
objZelle.FormulaLocal = "=SUMME(A2:A4)"
```

Die Funktion SUMME arbeitet natürlich nur in der deutschsprachigen OOo-Version korrekt! Wenn Sie eine US-amerikanische Version haben, steht in der Zelle die Fehlermeldung: #Name?

Die Formel selbst kann mit `Formula` und `FormulaLocal` ausgelesen werden. Ihr Ergebnistyp (empty = 0, Value = 1, Text = 2) kann mit `FormulaResultType` abgefragt werden. Analog können Sie auch mit den beiden Methoden `getFormula()` und `setFormula(String)` arbeiten. Letztere verlangt die US-amerikanische Schreibweise, erstere liefert sie in US-amerikanischer Schreibweise.

Analog zu `getFormula()` und `setFormula(String)` können Sie mit den Methoden `getString()`, `setString(String)`, `getValue()`, `setValue(Double)`, `getType()` und `getError()` arbeiten. Letzterer liefert die Fehlernummer als Long-Wert.

5.3.2 Zellen einfügen, kopieren und verschieben

Neben der Möglichkeit zum direkten Ändern der Zellinhalte bietet Calc die Schnittstelle „com.sun.star.sheet.XRangeMovement“ zum Einfügen, Löschen, Kopieren und Verschieben von Zellen an. Sie stellt vier Methoden zur Verfügung.

Das Einfügen von Zellen übernimmt die Methode `insertCell`.

```
Sub Zellzugriff05

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objZellAdresse As New com.sun.star.table.CellRangeAddress

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)

   objZellAdresse.Sheet = 0
   objZellAdresse.StartColumn = 1
   objZellAdresse.StartRow = 1
```

```
    objZellAdresse.EndColumn = 2
    objZellAdresse.EndRow = 2
    objBlatt.insertCells(objZellAdresse, _
        com.sun.star.sheet.CellInsertMode.DOWN)

End Sub
```

Das Beispiel fügt im ersten Tabellenblatt (es trägt die Nummer 0) in der zweiten Spalte und Zeile (sie tragen jeweils die Nummer 1) einen zwei Zeilen hohen und zwei Spalten breiten Bereich ein. Um den dafür notwendigen Platz zu schaffen, werden darunter liegende Werte entsprechend nach unten geschoben.

Die Definition des einzufügenden Bereiches erfolgt über die Struktur `com.sun.star.table.CellRangeAddress`.

Tabelle 5.47 CellRangeAddress umfasst folgende Werte:

Wert	Bereich	Beschreibung
Sheet	Short	die Nummer des Tabellenblatts (die Nummerierung beginnt mit 0)
StartColumn	Long	die Spalte, an welcher der Bereich eingefügt werden soll (die Nummerierung beginnt mit 0)
StartRow	Long	die Zeile, an welcher der Bereich eingefügt werden soll (die Nummerierung beginnt mit 0)
EndColumn	Long	die Spalte, bis zu welcher der einzufügende Bereich reicht (die Nummerierung beginnt mit 0)
EndRow	Long	die Zeile, bis zu welcher der einzufügende Bereich reicht (die Nummerierung beginnt mit 0)

Die ausgefüllte CellRangeAddress-Struktur muss der Methode `insertCells` als erster Parameter übergeben werden. Der zweite Parameter von `insertCells` enthält einen Wert der Aufzählung `com.sun.star.sheet.CellInsertMode` und legt fest, was mit den Werten passieren soll, die sich an der Einfügeposition befinden.

Tabelle 5.48 CellInsertMode kennt folgenden Werte:

Wert	Beschreibung
NONE	die aktuellen Werte bleiben an ihrem Platz.
DOWN	die Zellen an und unterhalb der Einfügeposition werden nach unten verschoben.
RIGHT	die Zellen an und rechts neben der Einfügeposition werden nach rechts verschoben.
ROWS	die Zeilen ab der Einfügeposition werden nach unten verschoben.

COLUMNS	die Spalten ab der Einfügeposition werden nach rechts verschoben.

Das Gegenstück zu der Methode `insertCells` stellt die Methode `removeRange` dar. Sie erwartet ebenfalls eine CellRangeAddress-Struktur und löscht den darin definierten Bereich aus der Tabelle.

```
Sub Zellzugriff06

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objZellAdresse As New com.sun.star.table.CellRangeAddress

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)

   objZellAdresse.Sheet = 0
   objZellAdresse.StartColumn = 1
   objZellAdresse.StartRow = 1
   objZellAdresse.EndColumn = 2
   objZellAdresse.EndRow = 2
   objBlatt.removeRange(objZellAdresse, _
      com.sun.star.sheet.CellDeleteMode.UP)

End Sub
```

Das Beispiel entfernt die Zellen des Bereiches B2:C3 und verschiebt die darunter liegenden Zellen zwei Zeilen nach oben. Die Art der Verschiebung wird über einen Wert der Aufzählung `com.sun.star.sheet.CellDeleteMode` definiert.

Tabelle 5.49 CellDeleteMode hält die folgende Werte bereit:

Wert	Beschreibung
NONE	die aktuellen Werte bleiben an ihrem Platz.
UP	die Zellen an und unterhalb der Einfügeposition werden nach oben verschoben.
LEFT	die Zellen an und rechts neben der Einfügeposition werden nach links verschoben.
ROWS	die Zeilen ab der Einfügeposition werden nach oben verschoben.

COLUMNS	die Spalten ab der Einfügeposition werden nach links verschoben.

Neben den Methoden `insertCells` und `removeRange` bietet die Schnittstelle „XRangeMovement" zwei weitere Methoden an, die es erlauben, Zellen zu verschieben (`moveRange`) beziehungsweise zu kopieren (`copyRange`). Das folgende Beispiel verschiebt den Bereich B2:C3 an die Position A6:

```
Sub Zellzugriff07

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objZellAdresse As New com.sun.star.table.CellRangeAddress
   Dim objZelle As New com.sun.star.table.CellAddress

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
   objZellAdresse.Sheet = 0
   objZellAdresse.StartColumn = 1
   objZellAdresse.StartRow = 1
   objZellAdresse.EndColumn = 2
   objZellAdresse.EndRow = 2

   objZelle.Sheet = 0
   objZelle.Column = 0
   objZelle.Row = 5

   objBlatt.moveRange(objZelle, objZellAdresse)
End Sub
```

Neben der bereits bekannten Struktur „CellRangeAdress", die den zu verschiebenden Bereich angibt, erwartet `moveRange` eine Struktur `com.sun.star.table.CellAddress`, die den Ursprung des Verschiebungsziels definiert.

Tabelle 5.50 CellAddress hält folgende Felder bereit:

Wert	Bereich	Beschreibung

Sheet	Short	die Nummer des Tabellenblatts (die Nummerierung beginnt mit 0)
Column	Long	die Nummer der adressierten Spalte (die Nummerierung beginnt mit 0)
Row	Long	die Nummer der adressierten Zeile (die Nummerierung beginnt mit 0)

Die Werte in der Zielregion der Verschiebung werden durch die Methode `moveRange` immer ohne Nachfragen überschrieben. Ein Parameter für eine automatische Verschiebung wie bei `insertCells` und `removeRange` steht nicht zur Verfügung.

Analog zur Methode `moveRange` arbeitet die Methode `copyRange`. Als einzigen Unterschied verschiebt sie nicht etwa den adressierten Bereich, sondern kopiert ihn:

```
objBlatt.copyRange(objZelle, objZellAdresse)
```

5.3.3 Weitere Informationen der Zelle

Tabelle 5.51 Weitere Informationen über die Zelle liefern folgende Eigenschaften und Methoden:

Eigenschaft	Information
Position.X	die X-Position als Long-Wert
Position.Y	die Y-Position als Long-Wert
Size.Width	die Breite in 1/100 mm als Long-Wert
Size.Height	die Höhe in 1/100 mm als Long-Wert

5.3.4 Zellinhalte löschen

Die Methode `clearContents()` löscht die Inhalte einer Zelle. Das folgende Beispiel löscht die Notiz aus der Zelle A4 des zweiten Blattes:

```
ThisComponent.sheets(1).getCellRangebyName("A4"). _
   clearContents(com.sun.star.sheet.CellFlags.ANNOTATION)
```

Tabelle 5.52 Folgende Parameter stehen für clearContents zur Verfügung:

Parameter	Wert	Bedeutung
EDITATTR	256	alles löschen
STRING	4	Zeichenketten
VALUE	1	Zahlen
DATETIME	2	Datum & Zeit

Parameter	Wert	Bedeutung
FORMULA	16	Formeln
ANNOTATION	8	Notizen
HARDATTR	32	Formate
OBJECTS	128	Objekte

EDITATTR und der Wert 256 funktionieren nicht!

5.4 Formatierungen

Ein Tabellenblatt stellt eine Reihe von Eigenschaften und Methoden zum Formatieren von Zellen zur Verfügung. Alle im Folgenden genannten Eigenschaften lassen sich sowohl auf einzelne Zellen als auch auf größere Zellbereiche, Zeilen und Spalten anwenden.

5.4.1 Zellattribute: Schrift

Jede Zelle unterstützt den Services „com.sun.star.style.CharacterProperties“. Eine Zelle kann beispielsweise wie folgt formatiert werden:

Tabelle 5.53 Die wichtigsten Zellattribute:

Wert	Bereich	Beschreibung
CharFontName	String	Name der ausgewählten Schriftart
CharColor	Long	Farbwert des Textes
CharHeight	Float	Höhe der Zeichen in Punkt (pt)
CharUnderline	Enum	Art des Unterstriches (Konstanten gemäß com.sun.star.awt.FontUnderline: NONE, SINGLE, DOUBLE, DOTTED, DONTKNOW, DASH, LONGDASH, DASHDOT, DASHDOTDOT, SMALLWAVE, WAVE, DOUBLEWAVE, BOLD, BOLDDOTTED, BOLDDASH, BOLDLONGDASH, BOLDDASHDOT, BOLDDASHDOTDOT, BOLDWAVE)
CharUnderlineHasColor	Boolean	der Unterstrich hat eine Farbe
CharUnderlineColor	Long	Farbwert des Unterstrichs

Wert	Bereich	Beschreibung
CharWeight	Enum	Schriftstärke (Konstanten gemäß com.sun.star.awt.Font-Weight): DONTKNOW, THIN, ULTRALIGHT, LIGHT, SEMI-LIGHT, NORMAL, SEMIBOLD, BOLD, ULTRABOLD und BLACK)
CharContoured	Boolean	Kontur
CharShadowed	Boolean	Schatten
CharRelief	Integer	Relief
CharStrikeout	Integer	durchgestrichen
CharKeepTogether	Boolean	Unterdrückung des automatischen Zeilenumbruches
CharStyleName	String	Name der Zeichenvorlage

Als weitere Schriftattribute, die sich im Wesentlichen auf asiatische Zeichensätze beziehen, sind zu nennen:

CharCrossedOut, CharEmphasis, CharFont, CharFontCharSet, CharFontCharSetAsian, CharFont-CharSetComplex, CharFontFamily, CharFontFamilyAsian, CharFontFamilyComplex, CharFont-NameAsian, CharFontNameComplex, CharFontPitch, CharFontPitchAsian, CharFontPitchCom-plex, CharFontStyleName, CharFontStyleNameAsian, CharFontStyleNameComplex, Char-HeightAsian, CharHeightComplex, CharLocale, CharLocaleAsian, CharLocaleComplex, CharPo-sture, CharPostureAsian, CharPostureComplex, CharWeightAsian und CharWeightComplex.

Im folgenden Beispiel wird Text in eine Zelle geschrieben unf formatiert:

```
Sub Zellzugriff08
   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object
   Dim objZelle As Object
   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
   objZelle = objBlatt.getCellByPosition(0, 0)
```

```
    objZelle.String = "Dies ist nur ein Test!"

    objZelle.CharFontName = "Arial"
    objZelle.CharColor = RGB(0, 0, 255)
    objZelle.CharHeight = 44
    objZelle.CharUnderline = 6
    objZelle.CharUnderlineHasColor = True
    objZelle.CharUnderlineColor = RGB(255, 0, 0)
    objZelle.CharWeight = com.sun.star.awt.FontWeight.BOLD
    objZelle.CharContoured = True
    objZelle.CharShadowed = True
    objBlatt.Columns(0).OptimalWidth = True
    objBlatt.Rows(0).OptimalHeight = True
End Sub
```

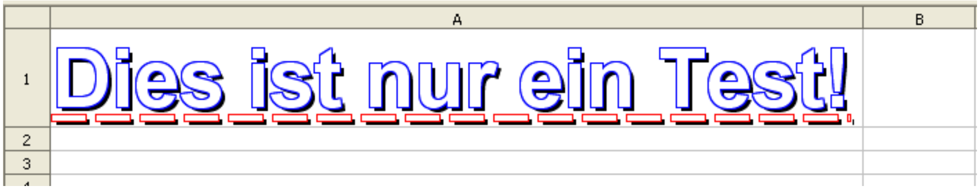


Abbildung 5.78 Text und Format

Da die genannten Schnittstellen einige Besonderheiten von Tabellenblättern außer Acht lassen, stellt OOo mit „com.sun.star.table.CellProperties“ einen weiteren Service zur Verfügung, der spezielle Eigenschaften für die Formatierung von Tabellenzellen bereithält. Die zentralen Eigenschaften dieses Services werden in den folgenden Abschnitten vorgestellt.

5.4.2 Zellattribute: Hintergrundfarbe und Schatten

Tabelle 5.54 Der Service com.sun.star.table.CellProperties stellt folgende Eigenschaften für den Umgang mit Hintergrundfarben und Schatten:

Wert	Wert	Beschreibung
CellBackColor	Long	Hintergrundfarbe der Tabellenzelle

Wert	Wert	Beschreibung
IsCellBackgroundTransparent	Boolean	setzt die Hintergrundfarbe transparent
ShadowFormat	Struct	Angaben zum Schatten der Tabellenzelle (Struktur gemäß com.sun.star.table.ShadowFormat)

Tabelle 5.55 Die Struktur com.sun.star.table.ShadowFormat mit den Detailangaben für den Tabellenschatten hat folgenden Aufbau:

Wert	Wert	Beschreibung
Location	Enum	Position des Schattens (Wert aus der Struktur com.sun.star.table.ShadowLocation: NONE, TOP_LEFT, TOP_RIGHT, BOTTOM_LEFT, BOTTOM_RIGHT)
ShadowWidth	Short	Größe des Schattens in 100stel Millimeter
IsTransparent	Boolean	setzt den Schatten transparent
Color	Long	Farbwert des Schattens

Das folgende Beispiel schreibt einen Text in die Zelle A1, ändert die Hintergrundfarbe über die Eigenschaft `CellBackColor` auf Gelb und legt einen hellblauen Schatten für die Tabellenzelle an, der um 1 mm nach rechts unten verschoben ist.

```
Sub Zellzugriff09

   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object
   Dim objZelle As Object
   Dim objSchattenFormat As New com.sun.star.table.ShadowFormat

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
   objZelle = objBlatt.getCellByPosition(0, 0)

   objZelle.String = "Dies ist ein weiterer Test!"
   objZelle.CharHeight = 44
   objZelle.CellBackColor = RGB(255, 255, 0)
```

```
    objSchattenFormat.Location = _
       com.sun.star.table.ShadowLocation.BOTTOM_RIGHT
    objSchattenFormat.ShadowWidth = 100
    objSchattenFormat.Color = RGB(0, 153, 255)
    objZelle.ShadowFormat = objSchattenFormat
    objBlatt.Columns(0).OptimalWidth = True
    objBlatt.Rows(0).OptimalHeight = True
End Sub
```

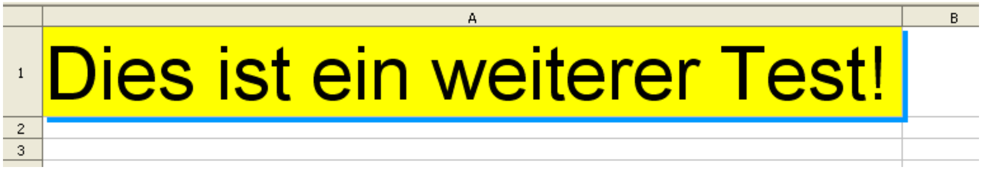


Abbildung 5.79 Die Zelle wird formatiert.

5.4.3 Zellattribute: Linien

Um einen Rahmen um eine Linie zu ziehen, können zwei verschiedene Varianten verwendet werden. Beide verwenden folgende Eigenschaften der Struktur `BorderLine`:

Tabelle 5.56 Die Eigenschaften der Struktur com.sun.star.table.BorderLine

Eigenschaft	**Beschreibung**
Color	Linienfarbe
InnerLineWidth	die Stärke der inneren Linie: 0 für keine, alle anderen Angaben in 0,01 mm
OuterLineWidth	die Stärke der äußeren Linie: 0 für keine, alle anderen Angaben in 0,01 mm
LineDistance	Abstand zwischen den Linien

Entweder Sie weisen diese Eigenschaften den Objekten „LeftBorder“, „RightBorder“, „TopBorder“ und „BottomBorder“ zu:

```
Sub MacheLinieUnten(objZelle As Object)
    Dim objLinie As New com.sun.star.table.BorderLine

    objLinie.Color = RGB(0, 0, 255)
```

```
   objLinie.OuterLineWidth = 10
   With objZelle
      .BottomBorder = objLinie
   End With
End Sub

Sub MacheLinieLinks(objZelle As Object)
   Dim objLinie As New com.sun.star.table.BorderLine

   objLinie.Color = RGB(0, 0, 255)
   objLinie.OuterLineWidth = 10
   With objZelle
      .LeftBorder = objLinie
   End With
End Sub
```

Oder Sie weisen es einer Eigenschaft des Objekts „TableBorder" zu. Es verfügt über folgende Eigenschaften:

Tabelle 5.57 Die Eigenschaften der Struktur com.sun.star.table.TableBorder:

Eigenschaft	Beschreibung
TopLine, BottomLine, LeftLine, RightLine, HorizontalLine, VerticalLine	Linienstil der oberen, unteren, linken, rechten, horizontal-mittleren und vertikal-mittleren Linie
IsTopLineValid, IsBottomLineValid, IsLeftLineValid, IsRightLineValid, IsHorizontalLineValid, IsVerticalLineValid	„True", falls die Linie eingeschaltet wurde
Distance	Abstand als short Integer Zahl
IsDistanceValid	„True", falls Abstand eingeschaltet wurde

Das Beispiel sieht dann wie folgt aus:

```
Sub MacheKasten(objZelle As Object)
   Dim objRand As New com.sun.star.table.TableBorder

   objRand.TopLine.Color = RGB(0, 0, 255)
   objRand.TopLine.InnerLineWidth = 10
   objRand.LeftLine.Color = RGB(0, 0, 255)
```

```
   objRand.LeftLine.InnerLineWidth = 10
   objRand.RightLine.Color = RGB(0, 0, 255)
   objRand.RightLine.InnerLineWidth = 10
   objRand.BottomLine.Color = RGB(0, 0, 255)
   objRand.BottomLine.InnerLineWidth = 10
   objRand.IsBottomLineValid = True

   objZelle.TableBorder = objRand

End Sub
```

Es wird beispielsweise wie folgt aufgerufen:

```
Sub TabelleFormatieren

   Dim objDatei as Object
   Dim objBlaetter As Object
   Dim objBlatt As Object
   Dim objZelle As Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
   objZelle = objBlatt.getCellRangeByName("A1:N1")
   Call MacheKasten(objZelle)
   objZelle = objBlatt.getCellRangeByName("A2")
   Call MacheKasten(objZelle)
   objZelle = objBlatt.getCellRangeByName("B2")
   Call MacheKasten(objZelle)
   objZelle = objBlatt.getCellRangeByName("C2:D2")
   Call MacheKasten(objZelle)
[...]

   objZelle = objBlatt.getCellRangeByName("A4:N4")
```

```
    Call MacheLinieUnten(objZelle)
    objZelle = objBlatt.getCellRangeByName("A5:A16")
    Call MacheLinieLinks(objZelle)
[...]

End Sub
```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1						Gemüse									
2	Die Angaben beziehen sich auf je 100 Gramm verzehrfertiges Nahrungsmittel	roh oder gekocht	kcal	kJ	% Kohlehydrate	Eiweiß	% Gesamtfett	% Wasser	% Ballaststoffe	mg Vitamin C	mg Calcium	mg Kalium	mg Phopsphor	mg Magnesium	
3															
4															
5	Artischocken	roh	57	239	11	+	3	86	3	9	52	410	110	26	
6	"	gek	57	239	11	+	3	86	3	6	50	315	90		
7	Auberginen	roh	25	105	5	+	1	93	3	5	16	210	26	11	
8	"	gek	16	67	3	+	1	93	3	5	16	210	26		
9	Blattsellerie	roh	20	84	4	+	1	95	2	9	45	290	40		
10	Blumenkohl	roh	28	117	8	+	2	92	2	76	24	380	60	7	
11	"	gek	20	84	3	+	2	94	2	45	18	250	52		
12	Bohnen, grün	gek	33	138	6	+	2	92	2	13	50	150	40	26	
13	Bohnen in Dosen	gek	24	100	4	+	1	93	3	4	35	145	25	20	
14	Bohnenkerne	gek	137	574	25	-	1	6	67		2	48	400	130	
15	Broccoli	roh	32	134	4	+	4	91	92	4	110	110	405	78	
16	"	gek	32	134	4	+	4	92		4	50	80	315	54	
17															
18															

Abbildung 5.80 Mache Linien …

Dieses Beispiel finden Sie in der Datei *TabelleFormatieren.sxc*. Ein weiteres Beispiel für Zellformatierungen finden Sie bei der Auswertung des Reporting-Tools (Kapitel 9).

5.4.4 Zellattribute: Ausrichtung

Calc stellt verschiedene Funktionen zur Verfügung, über die sich die Ausrichtung eines Textes innerhalb der Tabellenzelle ändern lässt.

Tabelle 5.58 Eigenschaften zur horizontalen und vertikalen Ausrichtung

Eigenschaft	Wert	Beschreibung
HoriJustify	Enum	Horizontale Ausrichtung des Textes (Wert aus com.sun.star.table.CellHoriJustify: STANDARD, LEFT, CENTER, RIGHT, BLOCK und REPEAT)
VertJustify	Enum	Vertikale Ausrichtung des Textes (Wert aus com.sun.star.table.CellVertJustify: STANDARD, TOP, CENTER und BOTTOM)
Orientation	Enum	Laufrichtung des Textes (Wert gemäß com.sun.star.table.CellOrientation: STANDARD, TOPBOTTOM, BOTTOMTOP und STACKED)
IsTextWrapped	Boolean	erlaubt einen automatischen Umbruch innerhalb der Zelle
RotateAngle	Long	Drehwinkel des Textes in 100stel Grad

Das folgende Beispiel zeigt, wie sich der Inhalt einer Tabellenzelle in der Mitte am unteren Ende Zelle darstellen lässt. Der Text wird zur besseren Lesbarkeit um 90 Grad gedreht.

```
Sub Zellzugriff10

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objZelle As Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
   objZelle = objBlatt.getCellByPosition(0, 0)

   objZelle.String = "Ich bin eine lange Überschrift"

   objZelle.HoriJustify = com.sun.star.table.CellHoriJustify.CENTER
   objZelle.VertJustify = com.sun.star.table.CellVertJustify.BOTTOM
   objZelle.Orientation = com.sun.star.table.CellOrientation.STANDARD
   objZelle.RotateAngle = 9000

   objBlatt.Columns(0).OptimalWidth = True
   objBlatt.Rows(0).OptimalHeight = True
```

```
End Sub
```

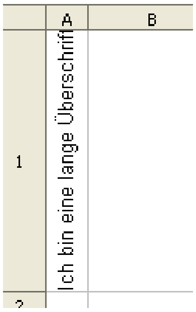


Abbildung 5.81 Die Ausrichtung des Textes wird geändert.

5.4.5 Zellschutz

Ein Zellschutz wird mit der Eigenschaft `CellProtection` aktiviert. Die Struktur `com.sun.star.util.CellProtection` verfügt über folgende Eigenschaften:

Tabelle 5.59 Eigenschaften des Zellschutzes

Eigenschaft	Beschreibung
IsLocked	gesperrt
IsFormulaHidden	die Formel wird ausgeblendet
IsHidden	vor dem Benutzer verborgen
IsPrintHidden	wird für den Ausdruck ausgeblendet

5.4.6 Zellattribute: Zahlen-, Datums- und Textformat

OOo bietet eine ganze Reihe von vordefinierten Datums- und Uhrzeitformaten. Jedes dieser Formate hat eine interne Nummer. Ist die Nummer bekannt, so lässt sich das betreffende Format einer Tabellenzelle über die Eigenschaft `NumberFormat` zuweisen. Da die Nummer eine interne Größe ist, ist es jedoch unmöglich, ohne weitere Hilfsmittel ein passendes Format auszuwählen. OOo stellt daher ein Objekt zur Verfügung, das Zugriff auf die existierenden Zahlenformate gewährt und eine Möglichkeit bietet, neue Zahlenformate anzulegen. Der Zugriff auf dieses Objekt erfolgt über den Aufruf:

```
NumberFormats = objDoc.NumberFormats
```

Seine Methoden `queryKey` und `addNew` gestatten die Suche nach vorhandenen Formaten und bieten die Möglichkeit, neue Formate hinzuzufügen.

Die Angabe eines Formats erfolgt über einen Format-String, der ähnlich dem der Format-Funktion von Calc aufgebaut ist. Es existiert jedoch ein wesentlicher Unterschied: Während der Format-Befehl immer englische Abkürzungen und Dezimal- beziehungsweise Tausender-Trennzeichen erwartet, müssen beim Aufbau eines Format-Befehls für das NumberFormats-Objekt die landesspezifischen Abkürzungen verwendet werden.

Das folgende Beispiel formatiert die Tabellenzelle A1 so, dass ihr Inhalt mit drei Nachkommastellen und einem Tausender-Trennzeichen dargestellt wird. Dabei erfolgt die Darstellung in Englisch unter Berücksichtigung der amerikanischen Landeseinstellungen.

```
Sub Zellzugriff11
   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objZelle As Object
   Dim objNummerFormat As Object
   Dim strNummerFormat As String
   Dim lngNummerFormatId As Long
   Dim objLocalSettings As New com.sun.star.lang.Locale

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
   objZelle = objBlatt.getCellByPosition(0, 0)

   objZelle.String = "Ich bin eine lange Überschrift"

   objZelle.Value = 23400.3523565
   objLocalSettings.Language = "en"
   objLocalSettings.Country = "us"
   objNummerFormat = objDatei.NumberFormats
   strNummerFormat = "#,##0.000"
   lngNummerFormatId = objNummerFormat.queryKey(strNummerFormat, _
      objLocalSettings, True)
   If lngNummerFormatId = -1 Then
```

```
      lngNummerFormatId = objNummerFormat.addNew(strNummerFormat, _
         objLocalSettings)
   End If
   objZelle.NumberFormat = lngNumberFormatId

End Sub
```

Abbildung 5.82 Eine Zahl wird formatiert.

Eine Erläuterung zum Code: Die Zeile

```
   lngNummerFormatId = objNummerFormat.queryKey(strNummerFormat, _
      objLocalSettings, True)
```

generiert die Nummer, hinter der sich das Zahlenformat verbirgt.

Einen Überblick zu den verschiedenen Formatierungsmöglichkeiten, liefert die Benutzerschnittstelle von OOo im Dialog zur Auswahl der Formatierungsinformationen.

Tabelle 5.60 Die wichtigsten Eigenschaft LocalSetting.Language:

Sprache	Kürzel	Sprache	Kürzel	Sprache	Kürzel
Afrikaans	af	French	fr	Polish	pl
Arabic	ar	Hebrew	he	Portuguese	pt
Catalan	ca	Croatian	hr	Russian	ru
Corsican	co	Hungarian	hu	Slovak	sk
Czech	cs	Indonesian	id	Slovenian	sl
Danish	da	Icelandic	is	Swedish	sv
German	de	Italian	it	Thai	th
Greek	el	Japanese	ja	Turkish	tr
English	en	Korean	ko	Vietnamese	vi
Persian	fa	Dutch	nl	Chinese	zh
Finnish	fi	Norwegian	no		

(Quelle: http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt – die Originalliste enthält 140 Sprachen! Die Datei befindet sich auf der Buch-CD im Ordner *Kapitel 5*.)

Tabelle 5.61 Die wichtigsten Eigenschaften von LocalSetting.Country

Land	**Kürzel**	**Land**	**Kürzel**	**Land**	**Kürzel**
ARGENTINA	AR	HUNGARY	HU	SLOVAKIA (Slovak Republic)	SK
AUSTRALIA	AU	ICELAND	IS	SLOVENIA	SI
AUSTRIA	AT	INDIA	IN	SOUTH AFRICA	ZA
BELGIUM	BE	INDONESIA	ID	SPAIN	ES
BOSNIA AND HERZEGOWINA	BA	IRAN (ISLAMIC REPUBLIC OF)	IR	SWEDEN	SE
BRAZIL	BR	IRAQ	IQ	SWITZERLAND	CH
CANADA	CA	IRELAND	IE	SYRIAN ARAB REPUBLIC	SY
CHILE	CL	ISRAEL	IL	TAIWAN	TW
CHINA	CN	ITALY	IT	THAILAND	TH
CROATIA (local name: Hrvatska)	HR	JAPAN	JP	TURKEY	TR
CUBA	CU	LUXEMBOURG	LU	UKRAINE	UA
CZECH REPUBLIC	CZ	MEXICO	MX	UNITED ARAB EMIRATES	AE
DENMARK	DK	NETHERLANDS	NL	UNITED KINGDOM	GB
EGYPT	EG	NORWAY	NO	UNITED STATES	US
ESTONIA	EE	POLAND	PL	VIET NAM	VN
FINLAND	FI	PORTUGAL	PT		
FRANCE	FR	RUSSIAN FEDERATION	RU		
GERMANY	DE	SAUDI ARABIA	SA		

(Quelle: http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html – die Originalliste enthält 250 Länder! Eine Liste befindet sich auf der Buch-CD im Ordner *Kapitel 5*)

Tabelle 5.62 Leichter funktioniert es durch Zuweisen des Numberfomats

Numberformat	Code	Beispiel
0	Standard	1234,12
1	0	1234
2	0,00	1234,12
3	#.##0	1.234
4	#.##0,00	1.234,12
5	#.###,00	1.234,12
10	0%	16%
11	0,00%	16,00%
108	#.##0 [$€-407];-#.##0 [$€-407]	1.234 €
109	#.##0,00 [$€-407];-#.##0,00 [$€-407]	1.234,12 €
110	#.##0 [$€-407];[ROT]-#.##0 [$€-407]	1.234 €
106	#.##0,00 [$€-407];[ROT]-#.##0,00 [$€-407]	1.234,12 €
111	#.##0,-- [$€-407];[ROT]-#.##0,-- [$€-407]	1.234,12 €
112	#.##0,00 [$EUR];-#.##0,00 [$EUR]	1.234,12 EUR
113	#.##0,00 [$EUR];[ROT]-#.##0,00 [$EUR]	1.234,12 EUR
30	TT.MM.JJ	01.04.05
38	NNNNT. MMMM JJJJ	Freitag, 1. April 2005
37	TT.MM.JJ	01.04.05
36	TT.MM.JJJJ	01.04.2005
39	T. MMM JJ	1. Apr 05
75	T. MMM JJJJ	1. Apr 2005
80	T. MMM. JJJJ	1. Apr. 2005
76	T. MMMM JJJJ	1. April 2005
81	T. MMMM JJJJ	1. April 2005
77	NN, T. MMM JJ	Fr, 1. Apr 05
31	NN TT.MMM JJ	Fr 01.Apr 05
78	NN, T. MMMM JJJJ	Fr, 1. April 2005
79	NNNNT. MMMM JJJJ	Freitag 1, April 2005
82	MM-TT	04-01
83	JJ-MM-TT	05-04-01
84	JJJJ-MM-TT	2005-04-01
32	MM.JJ	04.05

Numberformat	Code	Beispiel
33	TT.MMM	01.Apr
34	MMMM	April
35	QQ JJ	2. Quartal 05
85	WW	13
50	TT.MM.JJ HH:MM	01.04.05 00:00
51	TT.MM.JJJJ HH:MM:SS	01.04.05 00:00:00
40	HH:MM	09:45
41	HH:MM:SS	09:45:00
42	HH:MM AM/PM	09:45 vorm.
43	HH:MM:SS AM/PM	09:45:00 vorm.
44	[HH]:MM:SS	09:45:00
45	MM:SS,00	45:00,00
46	[HH]:MM:SS,00	09:45:00,00
50	TT.MM.JJ HH:MM	30.12.99 09:45
51	TT.MM.JJJJ HH:MM:SS	30.12.99 09:45:00
60	0,00E+000	1,23E+016
61	0,00E+00	1,23E+16
70	# ?/?	3 1/7
71	# ??/??	3 14/99
99	LOGISCH	WAHR
100	@	007
114	0,00 "kg"	123 kg
115	#.##0,00 "m²"	4.500,00 m²
und so weiter		

5.4.7 Formatierungen finden

Sie können nicht nur Zellen formatieren, Sie können über eine Schleife alle Bereiche ermitteln lassen, die unterschiedliche Formate tragen. Dabei hilft die Sammlung `getCellFormatRanges`.

```
Sub Zellzugriff12

   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objBereiche As Object
   Dim i As Integer
   Dim strBereiche As String

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(0)
   objBereiche = objBlatt.getCellFormatRanges()
   For i = 0 to objBereiche.Count -1
      strBereiche = strBereiche & Chr(13) & _
        objBereiche(i).getRangeAddress().StartRow() & "/" & _
        objBereiche(i).getRangeAddress().EndRow & "//" & _
        objBereiche(i).getRangeAddress().StartColumn() & "/" & _
        objBereiche(i).getRangeAddress().EndColumn
   Next
   Msgbox strBereiche

End Sub
```

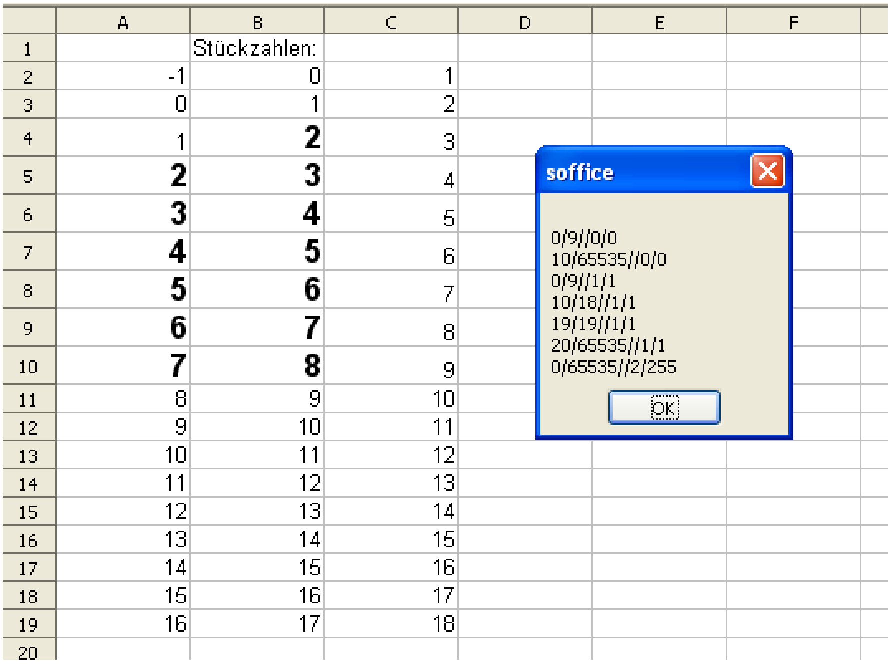


Abbildung 5.83 Verschieden formatierte Bereiche werden ausgelesen.

5.4.8 Zellen verbinden

Ein Bereich wird mit der Methode `Merge` zusammengefasst. Die Methode verlangt einen booleschen Parameter, der angibt, ob die Zellenzusammenfassung festgelegt oder aufgehoben wird:

```
objZelle = objBlatt.getCellRangebyPosition(1, 0, 7, 0)

objZelle.merge(True)
```

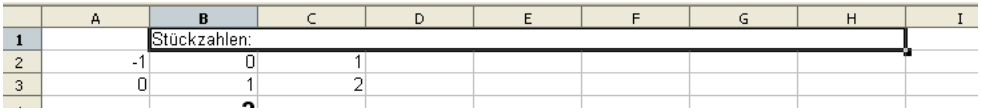


Abbildung 5.84 Zellen werden verbunden.

5.5 Weitere Attribute und Einstellungen von Zellen

5.5.1 Gültigkeit

Im Menü DATEN | GÜLTIGKEIT kann eine Gültigkeitsregel eingestellt werden. Damit kann die Eingabe des Benutzers auf bestimmte Datentypen (Zahlen, Datumsangaben, Zeitwerte, Texte einer bestimmten Länge) beschränkt werden:

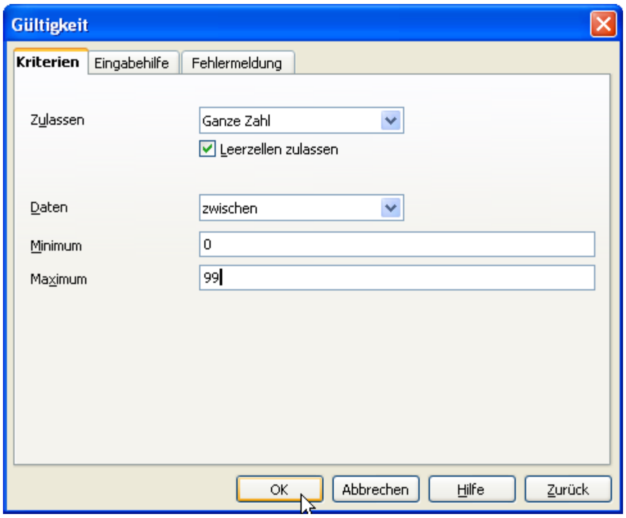


Abbildung 5.85 Daten | Gültigkeit

Dies kann programmiert werden:

```
Sub Zellzugriff13
   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objBereich As Object
   Dim objGueltigkeit As Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(3)
   objBereich = objBlatt.getCellRangeByName("B1")
```

```
   objBereich.String = "Stückzahlen:"
   objBereich = objBlatt.getCellRangeByName("B2:B20")

   objGueltigkeit  = objBereich.Validation
   With objGueltigkeit
      .Type = com.sun.star.sheet.ValidationType.WHOLE
      .setOperator(com.sun.star.sheet.ConditionOperator.BETWEEN)
      .setFormula1(0)
      .setFormula2(99)

      .InputTitle = "Hinweis"
      .InputMessage = "Bitte nur ganze Zahlen " & _
         "(zwischen 0 und 99) eingeben!"
      .ShowInputMessage = True

      .ErrorMessage = "Können Sie nicht lesen?" & Chr(13) & _
         "Bitte nur Zahlen eingeben!"
      .ErrorTitle = "Achtung!"
      .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
      .ShowErrorMessage = True
   End With

   objBereich.Validation = objGueltigkeit
End Sub
```

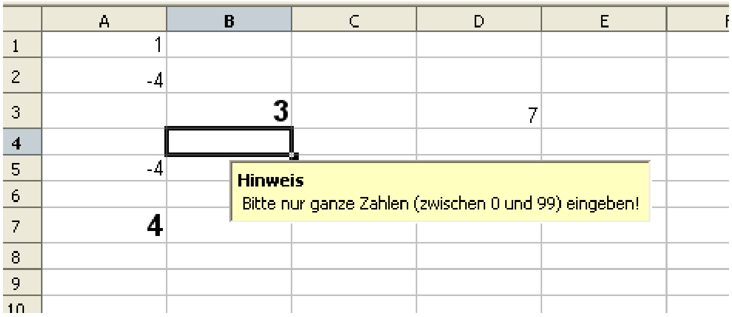


Abbildung 5.86 Die Gültigkeit der Zellen wird programmiert.

Tabelle 5.63 Das Objekt Validation besitzt folgende Methoden:

Methode	Rückgabewert	Beschreibung
getOperator ()	Long	liefert den gesetzten Operator
setOperator(Long)	Void	setzt den Operator
getFormula1()	String	liefert die Formel
getFormula2()	String	liefert die zweite Formel, falls vorhanden
setFormula1(String)	Void	setzt die Formel
setFormula2(String)	Void	setzt die zweite Formel, falls vorhanden
getSourcePosition()	Object	liefert die Zelladresse
setSourcePosition(Object)	Void	gibt die Zelladresse zurück
getPropertyValue(String)	Void	liefert die Eigenschaft
setPropertyValue(String, Variant)	Void	setzt die Eigenschaft

Tabelle 5.64 Das Objekt Validation besitzt folgende Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
ErrorAlertStyle	Long	Meldungsstil der Aufzählung „com.sun.star.ValidationAlertSty-le“: STOP, WARNING, INFO, MACRO
ErrorMessage	String	Meldungstext der Fehlermeldung
ErrorTitle	String	Titeltext der Fehlermeldung
IgnoreBlankCells	Boolean	ignoriere leere Zellen
InputMessage	String	Text der Eingabehilfe
InputTitle	String	Titel der Eingabehilfe
ShowErrorMessage	Boolean	zeige Fehlermeldung an
ShowInputMessage	Boolean	zeige Eingabehilfe an
Type	Long	Gültigkeitskriterium der Aufzählung „com.sun.star.sheet.Valida-tionType“: ANY, WHOLE, DECIMAL, DATE, TIME, TEXT_LEN, LIST, CU-STOM
Operator	Long	Vergleichsoperator der Aufzählung „com.sun.star.sheet.Conditi-onOperator“ NONE, EQUAL, NOT_EQUAL, GREATER, GREATER_EQUAL, LESS, LESS_EQUAL, BETWEEN,

Eigenschaft	Datentyp	Beschreibung
		NOT_BETWEEN, FORMULA
Formula1	String	Wert bei Vergleichsoperator
Formula2	String	zweiter Wert bei „zwischen“ oder „nicht zwischen“
SourcePosition	Object	Zelle oder Zellbereich, auf den die Gültigkeit angewandt wird

5.5.2 Bedingte Formatierung

Ähnlich wie die Gültigkeit für einen Bereich eingestellt wird, kann auch eine bedingte Formatierung aktiviert werden. Das benötigte Objekt heißt `ConditionalFormat`.

Tabelle 5.65 Folgende Methoden sind für `ConditionalFormat` interessant:

Methode	Beschreibung
addNew(Array)	fügt die Bedingungen hinzu
getCount()	liefert die Anzahl der festgelegten Bedingungen
clear()	löscht die Bedingungen
removeByIndex(Long)	löscht eine Bedingung
getElementNames()	liefert einen Array, in dem sich die Bedingungen befinden
hasByName(String)	überprüft, ob eine Bedingung eingeschaltet wurde.

Das folgende Beispiel formatiert im Zellbereich A1:B10 alle Zellen, deren Zahlenwert größer als 1 ist, mit dem Stil „Überschrift“:

```
Sub Zellzugriff14
   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objBereich As Object
   Dim objBedingung As Object
   Dim Bedingungen(2) As  new com.sun.star.beans.PropertyValue

   objDatei = ThisComponent
```

```
    objBlatt = objDatei.Sheets(3)
    objBereich = objBlatt.getCellRangeByName("A1:B10")

    Bedingungen(0).Name = "Operator"
    Bedingungen(0).Value = com.sun.star.sheet.ConditionOperator.GREATER

    Bedingungen(1).Name = "Formula1"
    Bedingungen(1).Value = "1"

    Bedingungen(2).Name = "StyleName"
    Bedingungen(2).Value = "Überschrift"

    objBedingung = objBereich.ConditionalFormat
    objBedingung.addNew(Bedingungen())
    objBereich.ConditionalFormat = objBedingung
End Sub
```

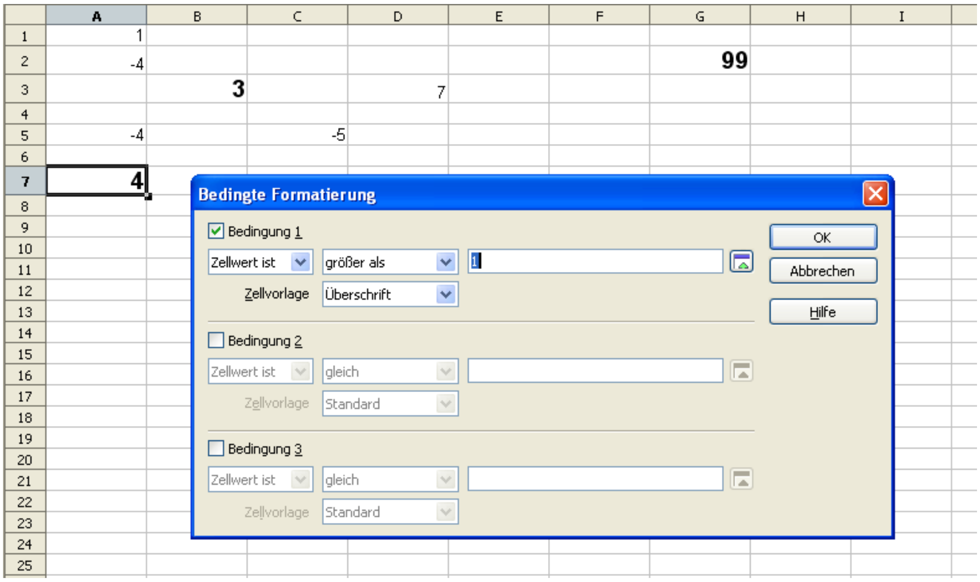


Abbildung 5.87 Die bedingte Formatierung

5.5.3 Notizen

Um eine Notiz in eine Zelle einzufügen, zu ändern oder zu löschen, steht das Annotation-Objekt zur Verfügung. Es besitzt einige Eigenschaften und Methoden, die sich sehr ähneln:

Tabelle 5.66 Wichtige Eigenschaften des Objekts Annotation:

Eigenschaft	Typ	Beschreibung
String	String	Text der Notiz
Author	String (kann nur gelesen werden)	Autor
Date	String (kann nur gelesen werden)	Erstelldatum
IsVisible	Boolean	sichtbar

Tabelle 5.67 Wichtige Methoden des Objekts Annotation:

Methode	Rückgabewert	Beschreibung
getString()	String	Text der Notiz
setString(String)	Void	Text der Notiz
insertString(Object, String, Boolean)	Void	Text der Notiz
getAuthor()	String	Autor
getDate()	String	Erstelldatum
getIsVisible()	Boolean	sichtbar
setIsVisible(Boolean)	Void	macht sichtbar

Das folgende Beispiel zeigt, wie in eine Zelle eine Notiz eingefügt wird. Sie wird danach sichtbar gemacht.

```
Sub Zellzugriff15


   Dim objDatei as Object

   Dim objBlatt As Object

   Dim objZelle As Object

   Dim objNotiz As Object


   objDatei = ThisComponent

   objBlatt = objDatei.Sheets(0)
```

```
   objZelle = objBlatt.getCellByPosition(0, 0)
   objNotiz = objZelle.Annotation
   objNotiz.String = "Ich bin ein überflüssiger Kommentar." _
      & Chr(13) & "Ich wurde erstellt von: " & _
      objNotiz.Author & Chr(13) & " am: " & _
      objNotiz.Date
   objNotiz.IsVisible = True
End Sub
```

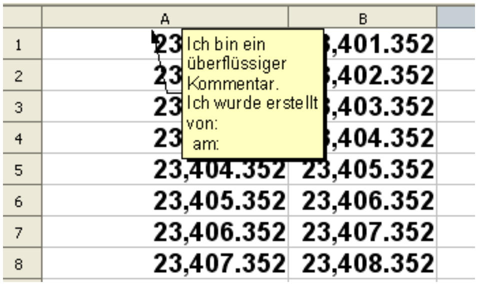


Abbildung 5.88 Ein Kommentar wird eingefügt.

Hier scheint ein Fehler vorzuliegen: Der Name des Autors und das Datum werden nicht ausgelesen.

Die Notiz kann folgendermaßen gelöscht werden:

```
      objZelle.clearContents(com.sun.star.sheet.CellFlags.ANNOTATION)
```

Sie können ausgedruckt werden, wenn in der Vorlage eines Seitenattributs die Eigenschaft `PrintAnnotations` gesetzt wird (siehe Kapitel 5.8.6). Mit der Sammlung „Annotations“ des Objektes „Sheet“ können alle Notizen durchlaufen werden:

```
   objNotizen = objBlatt.Annotations
   For i = 0 To objNotizen.Count -1
      Msgbox objNotizen(i).String
   Next
```

5.5.4 Datenreihen ausfüllen

Die Methode `fillSeries` füllt einen Datenbereich aus. Dabei werden die Richtung verlangt, der Reihentyp, die Zeiteinheit, der Endwert und das Inkrement.

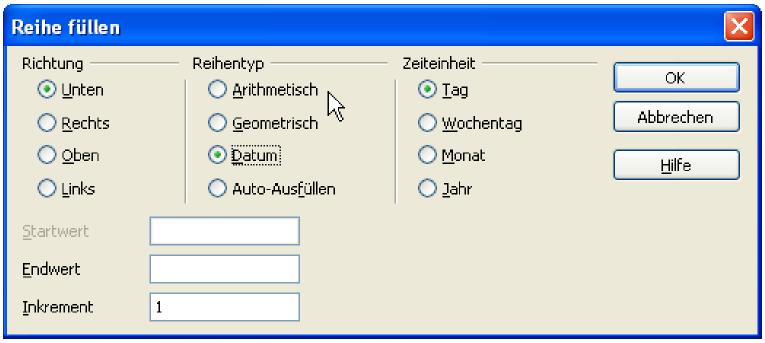


Abbildung 5.89 Reihe ausfüllen

Das folgende Beispiel schreibt in die Zelle A1 die Zahl 2 und füllt die Reihe bis H1 mit weiteren geraden Zahlen auf:

```
Sub Zellzugriff16
   Dim objDatei As Object
   Dim objBlatt As Object
   Dim objBereich as Object

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets.getByIndex(1)
   objBlatt.getCellRangebyName("A1").Value = 2
   objBereich = objBlatt.getCellRangebyName("A1:H1")

   objBereich.fillSeries(com.sun.star.sheet.FillDirection.TO_RIGHT, _
      com.sun.star.sheet.FillMode.LINEAR, _
      com.sun.star.sheet.FillDateMode.FILL_DATE_DAY, 2, 99)
End Sub
```

Analog können Sie Datumsangaben weiterzählen lassen:

```
   objBlatt.getCellRangebyName("A2").Value = DateValue("01.01.2005")
```

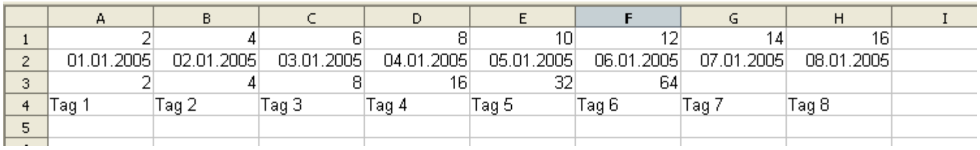
```
objBlatt.getCellRangebyName("A2").NumberFormat = 36
objBereich = objBlatt.getCellRangebyName("A2:H2")

objBereich.fillSeries(com.sun.star.sheet.FillDirection.TO_RIGHT, _
   com.sun.star.sheet.FillMode.LINEAR, _
   com.sun.star.sheet.FillDateMode.FILL_DATE_DAY, 1, 99999)
```

Oder Sie lassen geometrische Reihen ausfüllen:

```
objBlatt.getCellRangebyName("A3").Value = 2
objBereich = objBlatt.getCellRangebyName("A3:H3")
objBereich.fillSeries(com.sun.star.sheet.FillDirection.TO_RIGHT, _
   com.sun.star.sheet.FillMode.GROWTH, _
   com.sun.star.sheet.FillDateMode.FILL_DATE_DAY, 2, 99)
```

Oder Sie lassen Text-Zahl-Kombinationen weiterzählen:

```
objBlatt.getCellRangebyName("A4").String = "Tag 1"
objBereich = objBlatt.getCellRangebyName("A4:H4")
objBereich.fillSeries(com.sun.star.sheet.FillDirection.TO_RIGHT, _
   com.sun.star.sheet.FillMode.AUTO, _
   com.sun.star.sheet.FillDateMode.FILL_DATE_DAY, 1, 99)
```



	A	B	C	D	E	F	G	H	I
1	2	4	6	8	10	12	14	16	
2	01.01.2005	02.01.2005	03.01.2005	04.01.2005	05.01.2005	06.01.2005	07.01.2005	08.01.2005	
3	2	4	8	16	32	64			
4	Tag 1	Tag 2	Tag 3	Tag 4	Tag 5	Tag 6	Tag 7	Tag 8	
5									

Abbildung 5.90 Verschiedene Reihen

Wenn schon etwas in den Zellen steht, dann überschreibt die Methode `fillSeries` die alten Werte nicht. Zur Sicherheit sollten sie gelöscht werden, beispielsweise mit `clearContents()`.

Folgende Parameter stehen dabei zur Verfügung:

Die Konstanten von com.sun.star.sheet.FillDirection: TO_RIGHT, TO_LEFT, TO_TOP und TO_BOTTOM.

Tabelle 5.68 Die Konstanten der Aufzählung com.sun.star.sheet.FillMode:

Wert	Beschreibung
com.sun.star.sheet.FillMode.SIMPLE	konstante Reihe
com.sun.star.sheet.FillMode.LINEAR	lineare Reihe
com.sun.star.sheet.FillMode.GROWTH	geometrische Reihe
com.sun.star.sheet.FillMode.DATE	Datumsreihe
com.sun.star.sheet.FillMode.AUTO	benutzerdefinierte Liste

Tabelle 5.69 Für die Reihe DATE stehen folgende Konstanten zur Verfügung:

Wert	Beschreibung
com.sun.star.FillDateMode.FILL_DATE_DAY	zählt jeweils einen Tag weiter
com.sun.star.FillDateMode.FILL_DATE_WEEKDAY	zählt jeweils einen Wochentag weiter, d.h., Samstag und Sonntag werden übergangen
com.sun.star.FillDateMode.FILL_DATE_MONTH	zählt jeweils einen Monat weiter
com.sun.star.FillDateMode.FILL_DATE_YEAR	zählt jeweils ein Jahr weiter

5.6 Der Zellcursor

Wenn sichergestellt ist, dass auf einem Tabellenblatt ein zusammenhängender Bereich vorliegt (das heißt ohne Leerzeilen und ohne Leerspalten), dann kann die linke obere und rechte untere Zelle mit Hilfe des Zellcursors bestimmt werden.

Achtung: Anders als bei Writer kann bei Calc auf einem Tabellenblatt nur ein solcher Cursor existieren. Der Cursor hat nichts mit der sichtbaren Markierung des Tabellenblatts zu tun.

Der Cursor kann mit der Methode `createCursor()` oder besser mit `createCursorByRange(Zelle)` erzeugt werden. Die letztere Methode verlangt als Objekt eine Zelle. Im ersten Fall müsste man ihm eine Zelle oder einen Bereich mit den bekannten Methoden (`getCellByPosition`, `getCellRangeByPosition` oder `getCellRangeByName`) zuweisen.

Tabelle 5.70 Für den Zellcursor stehen folgende Methoden zur Verfügung:

Methode	Bedeutung
gotoStart()	„springt" zur linken, oberen Ecke
gotoEnd()	„springt" zur rechten, unteren Ecke
collapseToCurrentRegion()	dehnt den Cursor über den gesamten Bereich aus

Daneben stellt ein „ausgedehnter“ Cursor die Sammlung `Columns` und `Rows` zur Verfügung, die wiederum mit der Eigenschaft `Count` ihre Anzahl ermitteln lassen. Das folgende Beispiel zeigt mögliche Verwendungen der Zellcursors:

```
Sub ZellBereichErmitteln

   Dim objDatei As Object
   Dim objBlatt As Object
   Dim objZelle as Object
   Dim objCursor As Object
   Dim strInfo As String

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets.getByIndex(0)
   objZelle = objBlatt.getCellRangebyName("A1")

   objCursor = objBlatt.createCursorByRange(objZelle)
   objCursor.gotoEnd()
   strInfo = "Die untere Zeile liegt in Nr.: " & _
      objCursor.getRangeAddress().EndRow + 1 & Chr(13) & _
      "und die rechte Spalte liegt in Nr.: " & _
      objCursor.getRangeAddress().EndColumn
   MsgBox strInfo
   objCursor.gotoStart()
   strInfo = "Die obere Zeile liegt in Nr.: " & _
      objCursor.getRangeAddress().EndRow + 1 & Chr(13) & _
      "und die linke Spalte liegt in Nr.: " & _
      objCursor.getRangeAddress().EndColumn
   MsgBox strInfo
```

```
    objCursor = objBlatt.createCursorByRange(objZelle) '
    objCursor.collapseToCurrentRegion()
    strInfo = "Der Bereich verfügt über " & _
       objCursor.Columns().Count & _
       " Spalten " & Chr(13) & "und über " & _
       objCursor.Rows.Count & " Zeilen."
    MsgBox strInfo

End Sub
```

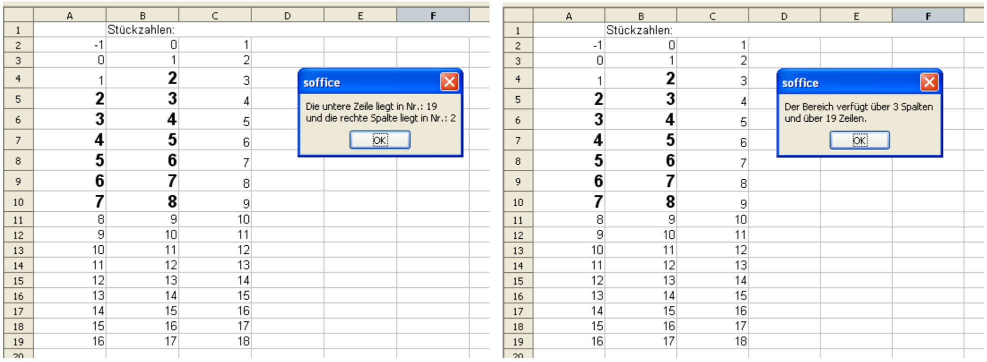


Abbildung 5.91 Der Bereich wird ermittelt.

Beachten Sie, dass `EndRow` und `EndColumn` bei 0 beginnen, also eine Zahl zurückgeben, die um 1 kleiner ist als der Wert, der durch die Eigenschaft `Count` ermittelt wird.

5.6.1 Auf welchem Blatt bin ich?

Um den Dateinamen der Datei auszulesen, die „oben“ in Calc liegt, kann das Objekt ThisComponent verwendet werden. Der Dateiname der „obersten“ Datei ist folglich:

```
MsgBox ThisComponent.URL
```

Um das selektierte Blatt zu ermitteln, kann wie folgt vorgegangen werden:

```
Dim objDoc As Object
Dim objBlatt As Object
```

```
Dim strName As String

objDoc = ThisComponent
objBlatt = objDoc.getCurrentController().getActiveSheet()
objBlatt = objDoc.getCurrentController().ActiveSheet
strName = objBlatt.Name
MsgBox strName
```

Dabei kann die Eigenschaft `ActiveSheet` oder die Methode `getActiveSheet()` verwendet werden.

5.6.2 In welcher Zelle bin ich?

Und so kann noch eine Ebene weiter nach unten „gewandert“ werden, um den markierten Zellbereich zu überprüfen. Da eine Zelle, mehrere Zellen oder mehrere getrennte Zellbereiche markiert sein können, muss dies überprüft werden:

```
Dim objDoc as Object
Dim objSel as Object
objDoc = ThisComponent
objSel = objDoc.getCurrentSelection()
If objSel.supportsService("com.sun.star.sheet.SheetCell" ) Then
   MsgBox "Eine einzelne Zelle ist selektiert"
Elseif objSel.supportsService("com.sun.star.sheet.SheetCellRange" ) Then
   MsgBox "Es wurde ein Zellbereich selektiert"
Elseif objSel.supportsService("com.sun.star.sheet.SheetCellRanges" ) Then
   MsgBox "Es wurden mehrere Zellbereiche selektiert"
Else
   MsgBox "Unbekannte Auswahl"
End If
```

Dieses Beispiel schlägt jedoch fehl, wenn ein Diagramm markiert ist!

```
Dim objSel as Object
```

```
Dim strText as String
Dim objBereich as Object
Dim i As Integer

objSel = ThisComponent.getCurrentSelection()
If objSel.supportsService("com.sun.star.sheet.SheetCellRanges") = _
   False Then
   MsgBox "keine Mehrfachselektion"
Else
   strText = "Es wurden mehrere Zellbereiche ausgewählt:" & Chr(13)
   strText = strText & "Anzahl der Bereiche: " & _
   objSel.getCount() & Chr(13) & Chr(13)

   For i = 0 To objSel.getCount() - 1
      objBereich = objSel.getByIndex(i).getRangeAddress()
      strText = strText & "Bereich " & i + 1 & ": " & _
         "Spalte " & objBereich.StartColumn + 1 & " - " & _
         "Zeile " & objBereich.StartRow + 1 & " bis " & _
         "Spalte " & objBereich.EndColumn + 1 &  " - " & _
         "Zeile " & objBereich.EndRow + 1 & Chr(13)
   Next
   MsgBox strText
End If
```

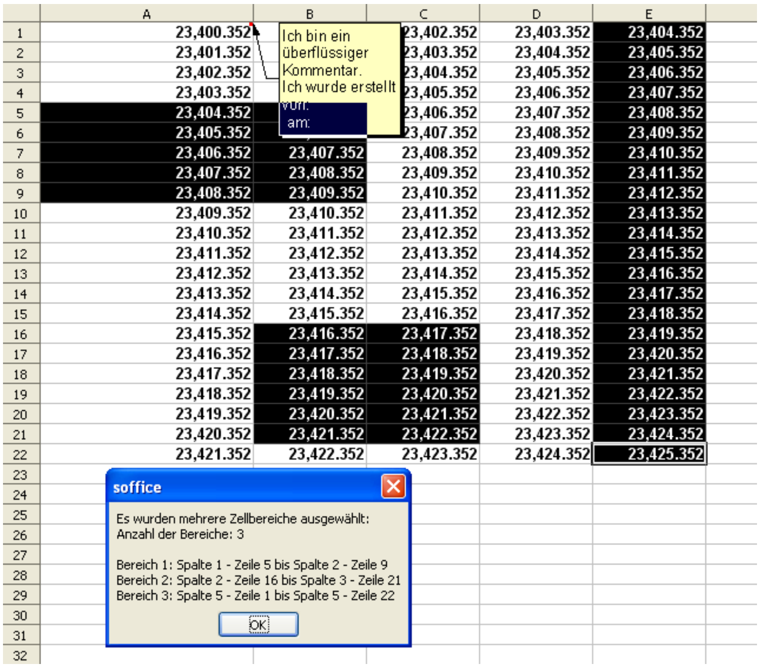


Abbildung 5.92 Die Bereiche der Mehrfachselektion werden bestimmt.

5.6.3 Daten trennen

Auch wenn OOo eine Reihe von Assistenten zur Verfügung stellt – ein wichtiger fehlt mir dennoch. Häufig geschieht es beim Importieren, dass mehrere Daten in einer Spalte zusammengefasst sind. Um sie vernünftig zu bearbeiten, müssen sie getrennt werden. Das folgende Beispiel überprüft, ob keine Mehrfachselektion aktiviert wurde, ob nur eine Spalte markiert wurde, und trennt anschließend die vorhandenen Daten nach dem vorgegebenen Trennzeichen in die nächsten Spalten:

```
Sub DatenTrennenStart
   Dim objSel as Object
   Dim objBereich as Object

   objSel = ThisComponent.getCurrentSelection()
   If objSel.supportsService("com.sun.star.sheet.SheetCellRanges") = _
```

```
      True Then
      MsgBox "Bitte keine Mehrfachselektion!"
   Else
      objBereich = objSel.getRangeAddress()
      If objBereich.StartColumn > objBereich.EndColumn Or _
         objBereich.StartColumn < objBereich.EndColumn Then
         MsgBox "Bitte nur eine Spalte markieren!"
      Else
         DialogLibraries.LoadLibrary("Standard")
         Dlg = CreateUnoDialog(DialogLibraries.Standard.dlgDatenTrennen)
         Dlg.Execute()
         ' -- der Dialog wird gestartet
      End If
   End If
End Sub
```

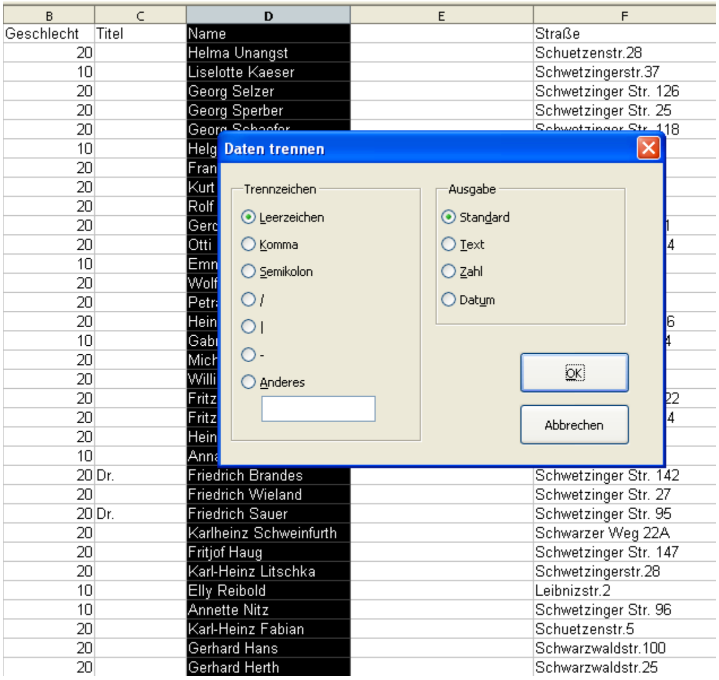


Abbildung 5.93 Der Auswahldialog wird gestartet.

Der Benutzer wählt ein vorgegebenes Trennzeichen oder gibt ein eigenes in die Liste ein. Bei der freien Eingabe wird überprüft, ob er genau ein Zeichen eingegeben hat:

```
Sub DatenTrennen
   Dim strTrennzeichen As String
   Dim intZaehler As Integer
   Dim intTeile As Integer
   Dim intSpaltenNr As Integer
   Dim intZeileOben As Integer
   Dim objSel As Object
   Dim objBereich As Object
   Dim objZelleAlt As Object
   Dim objZelleNeu As Object
   Dim strTeile() As String

   If Dlg.getControl("optAnderes").State = True  And _
      Dlg.getControl("txtTrennzeichen").getText() = "" Then
      MsgBox "Bitte geben Sie ein Trennzeichen ein!"
      Dlg.getControl("txtTrennzeichen").setFocus()
      Exit Sub
   ElseIf Dlg.getControl("optAnderes").State = True  And _
      Len(Dlg.getControl("txtTrennzeichen").getText()) > 1 Then
      MsgBox "Bitte geben Sie nur ein Trennzeichen ein!"
      Dlg.getControl("txtTrennzeichen").setFocus()
   End If

   If Dlg.getControl("optLeerzeichen").State = True Then
      strTrennzeichen = " "
   ElseIf Dlg.getControl("optKomma").State = True Then
      strTrennzeichen = ","
   ElseIf Dlg.getControl("optSemikolon").State = True Then
      strTrennzeichen = ";"
```

```
ElseIf Dlg.getControl("optSlash").State = True Then
   strTrennzeichen = "/"
ElseIf Dlg.getControl("optPipe").State = True Then
   strTrennzeichen = "|"
ElseIf Dlg.getControl("optBindestrich").State = True Then
   strTrennzeichen = "-"
ElseIf Dlg.getControl("optPipe").State = True Then
   strTrennzeichen = "|"
ElseIf Dlg.getControl("optAnderes").State = True Then
   strTrennzeichen = Dlg.getControl("txtTrennzeichen").getText()
End If
```

Da der Benutzer von oben nach unten oder von unten nach oben markieren kann, muss überprüft werden, welche Zelle die „oberste“ ist. Ebenso wird die Anzahl der Zeilen als Absolutwert genommen, um negative Zahlen zu vermeiden:

```
objSel = ThisComponent.getCurrentSelection()
objBereich = objSel.getRangeAddress()
intSpaltenNr = objBereich.EndColumn
If objBereich.StartRow < objBereich.EndRow Then
   intZeileOben = objBereich.StartRow
Else
   intZeileOben = objBereich.EndRow
End If
```

Und schließlich trennt die Funktion Split die Texte in jeder Zelle auf und schreibt das Ergebnis als Text, Zahl oder Datum in die Zellen danaben. Dabei darf natürlich nicht die Spaltenanzahl von 255 überschritten werden:

```
For intZaehler = 0 To Abs(objBereich.StartRow - objBereich.EndRow)
   objZelleAlt = _
      ThisComponent.getCurrentController().ActiveSheet. _
      getCellByPosition(intSpaltenNr, intZaehler + intZeileOben)
   strTeile() = Split(objZelleAlt.String, strTrennzeichen)
   For intTeile = 0 To Ubound(strTeile)
      If intSpaltenNr + intTeile >= 255 Then
         MsgBox "Ende des Tabellenblattes erreicht!"
```

```
            Exit Sub
         End If
         objZelleNeu = _
            ThisComponent.getCurrentController().ActiveSheet. _
            getCellByPosition(intSpaltenNr + intTeile, _
            intZaehler + intZeileOben)
         If Dlg.getControl("optZahl").State = True And _
            IsNumeric(strTeile(intTeile)) = True Then
            objZelleNeu.Value = strTeile(intTeile)
         ElseIf Dlg.getControl("optDatum").State = True And _
            IsDate(strTeile(intTeile)) = True Then
            objZelleNeu.Value = CDate(strTeile(intTeile))
            objZelleNeu.NumberFormat = 36
         Else
            objZelleNeu.String = strTeile(intTeile)
         End If
      Next
   Next
   Dlg.endExecute()
```

Abbildung 5.94 Einige Sekunden werden benötigt, um 4.000 Datensätze zu trennen.

Und natürlich funktioniert es auch mit Zahlen und Datumsangaben:

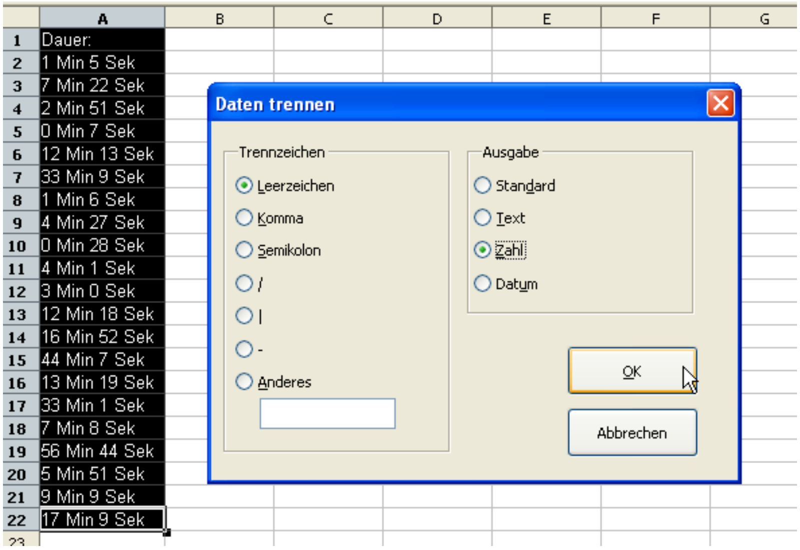


Abbildung 5.95 Die importierten Daten

	A	B	C	D	E	F	G
1	Dauer:						
2	1	Min	5	Sek			
3	7	Min	22	Sek			
4	2	Min	51	Sek			
5	0	Min	7	Sek			
6	12	Min	13	Sek			
7	33	Min	9	Sek			
8	1	Min	6	Sek			
9	4	Min	27	Sek			
10	0	Min	28	Sek			
11	4	Min	1	Sek			
12	3	Min	0	Sek			
13	12	Min	18	Sek			
14	16	Min	52	Sek			
15	44	Min	7	Sek			
16	13	Min	19	Sek			
17	33	Min	1	Sek			
18	7	Min	8	Sek			
19	56	Min	44	Sek			
20	5	Min	51	Sek			
21	9	Min	9	Sek			
22	17	Min	9	Sek			
23							

Abbildung 5.96 Die in Zahlen und Text getrennten Daten

5.7 Ein kleines Beispiel

An folgendem Beispiel (*FilmBsp*) soll das oben Beschriebene demonstriert werden. Auf einem Tabellenblatt befinden sich Informationen (Artikel, Preise und Artikelnummern). Über zwei Eingabefenster werden neue Artikel und ihre Preise eingegeben. Dazu wird die Anzahl der bereits vorhandenen Artikel ermittelt.

```
Sub FilmEintragen

   Dim objDatei As Object
   Dim objBlatt As Object
   Dim objZelle as Object
   Dim objCursor As Object
   Dim intZeilen As Integer
   Dim strFilmTitel As String
   Dim dblPreis As Double
```

```
   objDatei = ThisComponent
   objBlatt = objDatei.Sheets.getByIndex(0)
   objZelle = objBlatt.getCellRangebyName("A1")

   objCursor = objBlatt.createCursorByRange(objZelle)
   objCursor.collapseToCurrentRegion()
   intZeilen = objCursor.Rows.Count

   strFilmTitel = InputBox("Wie lautet der neue Film?")
   dblPreis = InputBox("Und was kostet " & strFilmtitel & "?")

   objZelle = objBlatt.getCellByPosition(0, intZeilen)
   objZelle.Value = objBlatt.getCellByPosition(0, intZeilen - _
      1).Value + 1
   objZelle = objBlatt.getCellByPosition(1, intZeilen)
   objZelle.String = strFilmTitel
   objZelle = objBlatt.getCellByPosition(2, intZeilen)
   objZelle.Value = dblPreis
   objZelle.NumberFormat = 106
End Sub
```

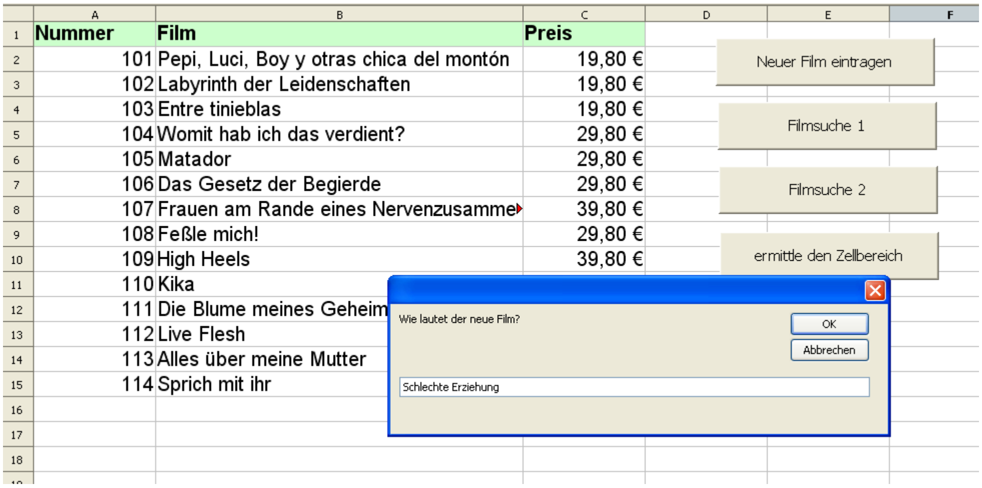


Abbildung 5.97 Ein neuer Artikel wird vom Benutzer eingegeben.

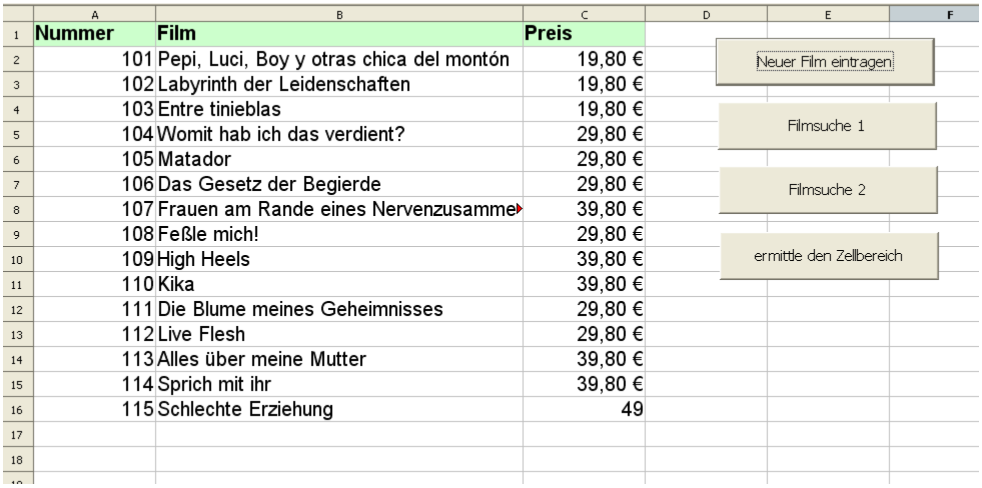


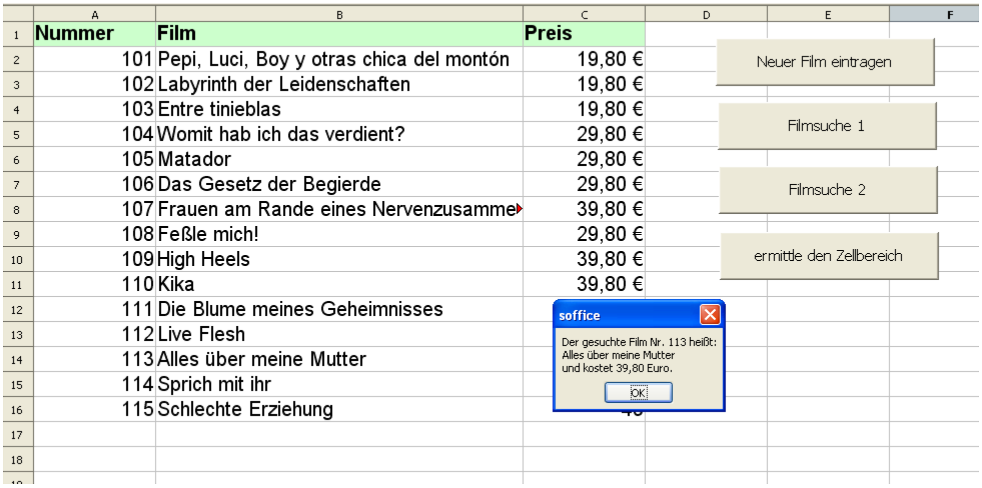
Abbildung 5.98 Der neue Artikel wird eingetragen.

Zur Erklärung: Der Zellcursor wird auf die Zelle A1 gesetzt, über den Bereich ausgedehnt und die Anzahl der Zeilen ermittelt. In die erste Zeile unterhalb des Bereiches wird die um 1 erhöhte Artikelnummer, der Film und der zugehörige Preis eingetragen.

Umgekehrt kann in dieser Liste nach einem Artikel über die Nummer der ersten Spalte gesucht werden:

```
Sub FilmSuchen1

   Dim objDatei As Object
   Dim objBlatt As Object
   Dim objZelle as Object
   Dim objCursor As Object
   Dim intZeilen As Integer
   Dim intZaehler As Integer
   Dim intNr As Integer

   objDatei = ThisComponent
   objBlatt = objDatei.Sheets.getByIndex(0)
   objZelle = objBlatt.getCellRangebyName("A1")

   objCursor = objBlatt.createCursorByRange(objZelle)
   objCursor.collapseToCurrentRegion()
   intZeilen = objCursor.Rows.Count

   intNr = InputBox("Wie lautet die gesuchte Nummer?")

   For intZaehler = 1 To intZeilen
      If objBlatt.getCellByPosition(0, intZaehler).Value = intNr Then
         Msgbox "Der gesuchte Film Nr. " & _
            intNr & " heißt:" & Chr(13) & _
         objBlatt.getCellByPosition(1, intZaehler).String & Chr(13) & _
         "und kostet " & _
         Format(objBlatt.getCellByPosition(2, intZaehler).Value, _
            "#,##0.00") & " Euro."
         Exit Sub
      End If
   Next
```

```
   MsgBox "Sorry, aber ein Artikel mit der Nr. " & intNr & " wurde nicht gefun-
den."
End Sub
```



Nummer	Film	Preis
101	Pepi, Luci, Boy y otras chica del montón	19,80 €
102	Labyrinth der Leidenschaften	19,80 €
103	Entre tinieblas	19,80 €
104	Womit hab ich das verdient?	29,80 €
105	Matador	29,80 €
106	Das Gesetz der Begierde	29,80 €
107	Frauen am Rande eines Nervenzusamme	39,80 €
108	Feßle mich!	29,80 €
109	High Heels	39,80 €
110	Kika	39,80 €
111	Die Blume meines Geheimnisses	
112	Live Flesh	
113	Alles über meine Mutter	
114	Sprich mit ihr	
115	Schlechte Erziehung	

Abbildung 5.99 Der Film wird gesucht und gefunden.

Eine Schleife durchläuft alle Zellen und überprüft ihren Inhalt mit dem gesuchten Wert. Wird kein Artikel gefunden, dann wird ein Hinweis ausgegeben. Oder analog kann nach einem Text gesucht werden. Dabei wird die Groß- und Kleinschreibung ausgeschaltet und eine Eingabe über einen Textteil ermöglicht:

```
Sub FilmSuchen2
[...]
   strTitel = InputBox("Wie lautet der gesuchte Titel " & Chr(13) & _
      "oder ein Teil davon?")

   For intZaehler = 1 To intZeilen
      If Instr(1, _
         LCase(objBlatt.getCellByPosition(1, intZaehler).String), _
         LCase(strTitel)) > 0 Then
         strListe = strListe & Chr(13) & _
            objBlatt.getCellByPosition(0, intZaehler).String
      End If
```

```
    Next

    If strListe = "" Then
       MsgBox "Sorry, aber ein Artikel mit dem Titel " & _
          strTitel & " wurde nicht gefunden."
    Else
       Msgbox "Folgende(r) Titel wurden gefunden:" & Chr(13) & _
          strListe
    End If
End Sub
```

5.8 Seitenattribute

Unter Seitenattributen versteht man sämtliche Formatierungsangaben, die sich auf die Positionierung des Dokumentinhaltes auf einer Seite beziehen sowie auf optische Elemente, die sich Seite für Seite wiederholen. Hierzu zählen unter anderem:

- das zugrunde liegende Papierformat,
- der Seitenrand
- die Kopf- und Fußzeilen

Das Vorgehen bei der Definition von Seitenattributen unterscheidet sich von dem anderer Formatierungen. Während der Anwender bei Zellen, Absätzen und Zeichenelementen die Wahl hat, ob er seine Formatierungen direkt an einem bestimmten Dokumentteil anwendet oder indirekt über den Einsatz von Vorlagen festlegt, können Seitenattribute prinzipiell nur über die zugehörige Seitenvorlage definiert werden. Beim Einfügen einer Kopf- oder Fußzeile muss daher beispielsweise nicht das eigentliche Dokument, sondern die verwendete Seitenvorlage um eine Kopf- beziehungsweise Fußzeile erweitert werden.

Nachfolgend findet sich eine Beschreibung der zentralen Attribute zur Formatierung von Dokumentseiten.

Ein großer Teil der beschriebenen Vorlagen steht nicht nur innerhalb von Tabellendokumenten zur Verfügung, sondern auch innerhalb von Textdokumenten. In dem Service „com.sun.star.style.PageProperties“ sind die für beide Dokumentarten gültigen Seitenattribute definiert. Diejenigen Attribute, die sich ausschließlich auf Tabellendokumente beziehen, finden sich im Service „com.sun.star.sheet.TablePageStyle“.

In OOo erfolgt die Definition der Seitenattribute über eine Seitenvorlage, die wiederum mit dem zugehörigen Dokument verknüpft wird.

5.8.1 Seitenhintergrund

Tabelle 5.71 Die Eigenschaften von com.sun.star.style.PageProperties für den Seitenhintergrund:

Eigenschaft	Wert	Beschreibung
BackColor	Long	Farbe des Hintergrunds
BackGraphicURL	String	URL der zu verwendenden Hintergrundgrafik
BackGraphicFilter	String	Name des Filters zur Interpretation der Hintergrundgrafik
BackGraphicLocation	Enum	Position der Hintergrundgrafik (Wert laut Aufzählung com.sun.star.style.GraphicLocation)
BackTransparent	Boolean	stellt den Hintergrund transparent dar

5.8.2 Seitenformat

Tabelle 5.72 Die Eigenschaften von com.sun.star.style.PageProperties für das Seitenformat:

Eigenschaft	Wert	Beschreibung
IsLandscape	Boolean	Ausgabe im Querformat
Width	Long	Breite der Seite in 100stel Millimetern
Height	Long	Höhe der Seite in 100stel Millimetern
PrinterPaperTray	String	Name des zu verwendenden Drucker-Ausgabeschachtes

Das folgende Beispiel setzt die Seitengröße der Seitenvorlage „Default“ auf die Werte für das Format DIN A5 quer (Höhe 14,8 cm, Breite 21 cm):

```
Sub SeiteEinrichten01
   Dim objDatei As Object
   Dim objStyleFamilies As Object
   Dim objSeitenvorlage As Object
   Dim objDefPage As Object
   objDatei = ThisComponent
```

```
    objStyleFamilies = objDatei.StyleFamilies
    objSeitenvorlage = objStyleFamilies.getByName("PageStyles")
    objDefPage = objSeitenvorlage.getByName("Default")
    objDefPage.IsLandscape = True
    objDefPage.Width = 21000
    objDefPage.Height = 14800

End Sub
```

5.8.3 Seitenrand, -rahmen und -schatten

Zur Anpassung des Seitenrandes sowie eventueller Rahmen und Schatten steht der Service „com.sun.star.style.PageProperties“ bereit.

Tabelle 5.73 com.sun.star.style.PageProperties verfügt über folgende Eigenschaften

Eigenschaft	Wert	Beschreibung
LeftMargin, RightMargin, TopMargin und BottomMargin	Long	linker, rechter, oberer und unterer Seitenrand in 100stel Millimetern
LeftBorder, RightBorder, TopBorder und BottomBorder	Struct	Angaben zur linken, rechten, oberen und unteren Linie des Seitenrahmens (Struktur com.sun.star.table.BorderLine: Color, InnerLineWidth, OuterLineWidth und LineDistance)
LeftBorderDistance, RightBorderDistance, TopBorderDistance und BottomBorderDistance	Long	Abstand zwischen linkem, rechtem, oberem und unterem Seitenrahmen und Seiteninhalt in 100stel Millimetern
ShadowFormat	Struct	Angaben zum Schatten des Content-Bereiches der Seite (Struktur com.sun.star.table.ShadowFormat: Location, ShadowWidth, IsTransparent und Color)

Das folgende Beispiel setzt den linken, rechten, oberen und unteren Rand der Seitenvorlage „Default“ auf 1 Zentimeter, und zeichnet um das Tabellenblatt eine doppelte rote Linie im Abstand von einem Millimeter.

```
Sub SeiteEinrichten02
    Dim objDatei As Object
    Dim objStyleFamilies As Object
```

```
   Dim objSeitenvorlage As Object
   Dim objDefPage As Object
   Dim objRandLinie As New com.sun.star.table.BorderLine

   objDatei = ThisComponent
   objStyleFamilies = objDatei.StyleFamilies
   objSeitenvorlage = objStyleFamilies.getByName("PageStyles")
   objDefPage = objSeitenvorlage.getByName("Default")
   objDefPage.LeftMargin = 2000
   objDefPage.RightMargin = 2000
   objDefPage.TopMargin = 2000
   objDefPage.BottomMargin = 2000

   objRandLinie.Color = RGB(255, 0, 0)
   objRandLinie.InnerLineWidth = 1
   objRandLinie.OuterLineWidth = 1
   objRandLinie.LineDistance = 100

   objDefPage.LeftBorder = objRandLinie
   objDefPage.RightBorder = objRandLinie
   objDefPage.TopBorder = objRandLinie
   objDefPage.BottomBorder = objRandLinie
End Sub
```

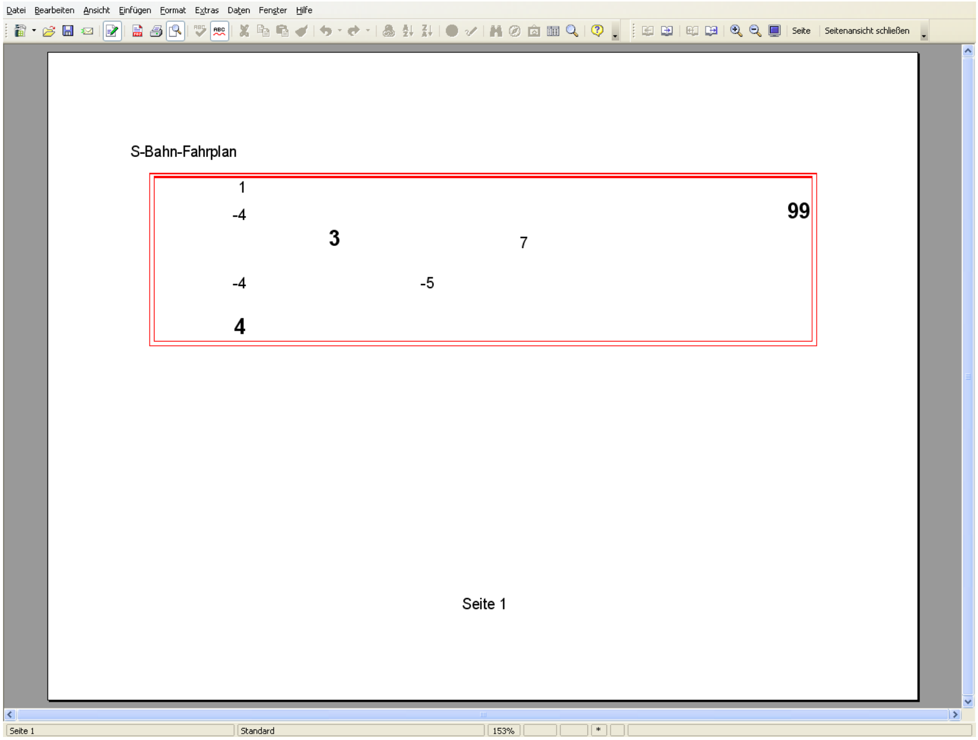


Abbildung 5.100 Die Seite wird eingerichtet.

5.8.4 Kopf- und Fußzeilen

Die Kopf- und Fußzeilen eines Dokumentes gehören zu den Seitenattributen und werden ebenfalls über den Service „com.sun.star.style.PageProperties“ definiert.

Tabelle 5.74 Die wichtigsten Eigenschaften für Kopfzeilen und Fußzeilen lauten:

Eigenschaft	Wert	Beschreibung
HeaderIsOn/FooterIsOn	Boolean	Kopfzeile/Fußzeile ist aktiviert
HeaderLeftMargin/FooterLeftMargin	Long	Abstand zwischen Kopfzeile/Fußzeile und linkem Seitenrand in 100stel Millimetern
HeaderRightMargin/FooterRightMargin	Long	Abstand zwischen Kopfzeile/Fußzeile und rechtem Seitenrand in 100stel Millimetern

Eigenschaft	Wert	Beschreibung
HeaderBodyDistance/FooterBodyDistance	Long	Abstand zwischen Kopfzeile/Fußzeile und Hauptbereich des Dokumentes in 100stel Millimetern
HeaderHeight/FooterHeight	Long	Höhe der Kopfzeile/Fußzeile in 100stel Millimeter
HeaderIsDynamicHeight/FooterIsDynamicHeight	Boolean	die Höhe der Kopfzeile/Fußzeile wird automatisch an den Inhalt angepasst
HeaderLeftBorder/FooterLeftBorder	Struct	Angaben zur linken Linie des Rahmens um die Kopfzeile/Fußzeile (Struktur com.sun.star.table.BorderLine: Color, InnerLineWidth, OuterLineWidth und LineDistance)
HeaderRightBorder/FooterRightBorder	Struct	Angaben zur rechten Linie des Rahmens um die Kopfzeile/Fußzeile (Struktur com.sun.star.table.BorderLine)
HeaderTopBorder/FooterTopBorder	Struct	Angaben zur oberen Linie des Rahmens um die Kopfzeile/Fußzeile (Struktur com.sun.star.table.BorderLine)
HeaderBottomBorder/FooterBottomBorder	Struct	Angaben zur unteren Linie des Rahmens um die Kopfzeile/Fußzeile (Struktur com.sun.star.table.BorderLine)
HeaderLeftBorderDistance	Long	Abstand zwischen linker Rahmenlinie und Inhalt der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderRightBorderDistance/ FooterRightBorderDistance	Long	Abstand zwischen rechter Rahmenlinie und Inhalt der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderTopBorderDistance/FooterTopBorderDistance	Long	Abstand zwischen oberer Rahmenlinie und Inhalt der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderBottomBorderDistance/ FooterBottomBorderDistance	Long	Abstand zwischen unterer Rahmenlinie und Inhalt der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderIsShared/FooterIsShared	Boolean	die Kopfzeilen/Fußzeilen auf den linken und rechten Seiten enthalten den gleichen Inhalt (siehe HeaderText, HeaderTextLeft und HeaderTextRight bzw. FooterText, FooterTextLeft und FooterTextRight)
HeaderBackColor/FooterBackColor	Long	Hintergrundfarbe der Kopfzeile/Fußzeile

Eigenschaft	Wert	Beschreibung
HeaderBackGraphicURL/ FooterBackGraphicURL	String	URL der zu verwendenden Hintergrundgrafik
HeaderBackGraphicFilter/FooterBackGraphicFilter	String	Name des Filters zur Interpretation der Hintergrundgrafik für die Kopfzeile/Fußzeile
HeaderBackGraphicLocation/ FooterBackGraphicLocation	Enum	Position der Hintergrundgrafik für die Kopfzeile/Fußzeile (Wert laut Aufzählung com.sun.star.style.GraphicLocation)
HeaderBackTransparent/FooterBackTransparent	Boolean	zeigt den Hintergrund der Kopfzeile/Fußzeile transparent an
HeaderShadowFormat/FooterShadowFormat	Struct	Angaben zum Schatten der Kopfzeile/ Fußzeile (Struktur com.sun.star.table.ShadowFormat)

5.8.5 Texte von Kopf- und Fußzeilen ändern

Beim Einfügen von Text in die Kopf- und Fußzeilen sind einige Besonderheiten zu beachten.

Tabelle 5.75 Der Zugriff auf die Kopf- und Fußzeilen eines Tabellendokumentes erfolgt über die Eigenschaften

Eigenschaft	Wert	Beschreibung
LeftPageHeaderContent, RightPageHeaderContent, LeftPageFooterContent und RightPageFooterContent	Object	Inhalt der Kopfzeilen und Fußzeilen für die linken oder rechten Seiten des Dokumentes (Service com.sun.star.sheet.HeaderFooterContent)

Ist keine Unterscheidung zwischen rechter und linker Kopf- beziehungsweise Fußzeile gewünscht (die Eigenschaft `FooterIsShared` ist „False“), so sind die Eigenschaften für die rechte Kopf- beziehungsweise Fußzeile zu verwenden.

Alle genannten Objekte liefern ein Objekt zurück, das den Service „com.sun.star.sheet.HeaderFooterContent“ unterstützt. Dieser Service stellt über die (unechten) Eigenschaften `LeftText`, `CenterText` und `RightText` drei Textelemente für die Kopf- und Fußzeilen von Calc zur Verfügung.

Das folgende Beispiel schreibt in das linke Textfeld der Kopfzeile aus der Vorlage „Default“ den Text „S-Bahn-Fahrplan“ (siehe Abbildung 5.32).

```
Sub SeiteEinrichten03

   Dim objDatei As Object
```

```
    Dim objStyleFamilies As Object
    Dim objSeitenvorlage As Object
    Dim objDefPage As Object
    Dim objHText As Object
    Dim objHInhalt As Object
    objDatei = ThisComponent
    objStyleFamilies = objDatei.StyleFamilies
    objSeitenvorlage = objStyleFamilies.getByName("PageStyles")
    objDefPage = objSeitenvorlage.getByName("Default")

    objDefPage.HeaderIsOn = True
    objHInhalt = objDefPage.RightPageHeaderContent
    objHText = objHInhalt.LeftText
    objHText.String = "S-Bahn-Fahrplan"
    objDefPage.RightPageHeaderContent = objHInhalt

End Sub
```

An dem Beispiel ist insbesondere die letzte Zeile zu beachten. Nach dem Ändern des Textes muss das TextContent-Objekt erneut der Kopfzeile zugewiesen werden, damit die Änderung wirksam wird.

5.8.6 Zentrierung und Festlegung der auszudruckenden Elemente

In den Seitenvorlagen von Calc kann festgelegt werden, ob der zu druckende Tabellenbereich auf der Seite zu zentrieren ist oder am linken, beziehungsweise oberen Rand steht. Dafür wird der Service „com.sun.star.sheet.TablePageStyle“ verwendet.

Tabelle 5.76 com.sun.star.sheet.TablePageStyle bietet folgende Eigenschaften an:

Eigenschaft	Wert	Beschreibung
CenterHorizontally	Boolean	Tabelleninhalt wird horizontal zentriert
CenterVertically	Boolean	Tabelleninhalt wird vertikal zentriert

Bei der Seitenformatierung von Tabellen kann der Anwender festlegen, welche Elemente einer Seite bei einem Ausdruck sichtbar sein sollen und welche nicht. Hierzu hält der Service „com.sun.star.sheet.TablePageStyle“ folgende Eigenschaften bereit:

Tabelle 5.77 Eigenschaften von com.sun.star.sheet.TablePageStyle

Eigenschaft	**Wert**	**Beschreibung**
PrintAnnotations	Boolean	druckt Zellanmerkungen mit aus
PrintGrid	Boolean	druckt das Gitternetz der Tabelle mit aus
PrintHeaders	Boolean	druckt die Zeilen- und Spaltentitel mit aus
PrintCharts	Boolean	druckt Diagramme innerhalb der Tabellen mit aus
PrintObjects	Boolean	druckt eingebettete Objekte mit aus
PrintDrawing	Boolean	druckt Zeichenelement-Objekte mit aus
PrintDownFirst	Boolean	druckt bei mehrseitigen Tabellen zuerst die unten anschließenden Seiten und dann die seitlich anschließenden Seiten aus
PrintFormulas	Boolean	druckt die Formeln statt der berechneten Werte aus
PrintZeroValues	Boolean	druckt Nullwerte mit aus

```
Sub SeiteEinrichten04
   Dim objDatei As Object
   Dim objStyleFamilies As Object
   Dim objSeitenvorlage As Object
   Dim objDefPage As Object
   Dim objHText As Object
   Dim objHInhalt As Object

   objDatei = ThisComponent
   objStyleFamilies = objDatei.StyleFamilies
   objSeitenvorlage = objStyleFamilies.getByName("PageStyles")
   objDefPage = objSeitenvorlage.getByName("Default")

   objDefPage.CenterHorizontally = True
   objDefPage.PrintAnnotations = True
```

```
    objDefPage.PrintGrid = True
    objDefPage.PrintDrawing = False
    objDefPage.PrintObjects = False
    objDefPage.PrintCharts = False
    objDefPage.PrintHeaders = True
    objDefPage.PrintDownFirst = True
    objDefPage.PrintFormulas = False
    objDefPage.PrintZeroValues = True

End Sub
```

5.9 Hilfsmittel in Calc

5.9.1 Sortieren

Um Daten zu sortieren, steht Ihnen in OOo-Basic die Methode `Sort` zur Verfügung.

```
Sub Sortieren1
    Dim objDatei as Object
    Dim objBlatt As Object
    Dim objBereich As Object
    Dim SortierKrit(3) As new com.sun.star.beans.PropertyValue
    Dim SortierFeld(0) As new com.sun.star.table.TableSortField

    objDatei = ThisComponent
    objBlatt = objDatei.Sheets(4)
    objBereich = objBlatt.getCellRangeByName("A1:F76")

    SortierFeld(0).Field = 0
    SortierFeld(0).IsAscending = True
    SortierFeld(0).FieldType = _
        com.sun.star.util.SortFieldType.ALPHANUMERIC
```

```
   SortierKrit(0).Name = "SortFields"
   SortierKrit(0).Value = SortierFeld()
   SortierKrit(1).Name = "SortColumns"
   SortierKrit(1).Value = False
   SortierKrit(2).Name = "ContainsHeader"
   SortierKrit(2).Value = true
   SortierKrit(3).Name = "IsCaseSensitive"
   SortierKrit(3).Value = false
   objBereich.Sort(SortierKrit())
End Sub
```

	A	B	C	D	E	F	G
1	Datum	Verkäufer	Artikel	Kunde	Menge	Umsatz	
2	01.01.04	C. Breuer	Klebeetikette	Papier & Dec	23	4.853,00 €	
3	02.01.04	C. Breuer	Briefpapier	Hugos Shop	12	780,00 €	
4	03.01.04	B. Weidner	Klebeetikette	Hugos Shop	10	1.900,00 €	
5	03.01.04	B. Weidner	Briefumschlä	Papier 2002	75	7.500,00 €	
6	03.01.04	B. Weidner	Briefpapier	Art & Design	30	1.650,00 €	
7	04.01.04	E. Sauerbier	Briefpapier	Art & Design	100	6.500,00 €	
8	04.01.04	E. Sauerbier	Klebeetikette	Papier 2002	10	1.800,00 €	
9	05.01.04	C. Breuer	Klebeetikette	Casarossa	15	4.650,00 €	
10	05.01.04	C. Breuer	Briefpapier	Hugos Shop	12	780,00 €	
11	06.01.04	B. Weidner	Briefpapier	Papier 2002	20	1.400,00 €	
12	06.01.04	B. Weidner	Briefumschlä	Hugos Shop	45	4.950,00 €	
13	09.01.04	E. Sauerbier	Klebeetikette	Hugos Shop	20	3.600,00 €	
14	09.01.04	E. Sauerbier	Briefumschlä	Art & Design	100	10.000,00 €	
15	10.01.04	C. Breuer	Briefpapier	Papier 2002	20	1.300,00 €	
16	11.01.04	B. Weidner	Klebeetikette	Casarossa	50	11.500,00 €	
17	11.01.04	B. Weidner	Briefumschlä	Papier & Dec	55	5.225,00 €	
18	12.01.04	E. Sauerbier	Briefumschlä	Papier 2002	95	9.975,00 €	
19	12.01.04	E. Sauerbier	Klebeetikette	Hugos Shop	45	9.900,00 €	
20	13.01.04	C. Breuer	Briefumschlä	Art & Design	150	18.000,00 €	
21	16.01.04	B. Weidner	Briefpapier	Casarossa	80	3.600,00 €	
22	17.01.04	E. Sauerbier	Klebeetikette	Uschi	12	2.160,00 €	
23	17.01.04	E. Sauerbier	Briefpapier	Uschi	400	26.000,00 €	
24	17.01.04	E. Sauerbier	Briefumschlä	Art & Design	75	7.500,00 €	
25	18.01.04	C. Breuer	Klebeetikette	Hugos Shop	30	6.000,00 €	
26	18.01.04	C. Breuer	Briefpapier	Casarossa	60	3.900,00 €	
27	19.01.04	B. Weidner	Briefumschlä	Papier & Dec	80	8.000,00 €	
28	19.01.04	B. Weidner	Briefpapier	Casarossa	100	7.500,00 €	
29	20.01.04	E. Sauerbier	Briefpapier	Hugos Shop	67	4.355,00 €	
30	20.01.04	E. Sauerbier	Klebeetikette	Papier & Dec	50	11.000,00 €	
31	23.01.04	C. Breuer	Klebeetikette	Art & Design	28	7.000,00 €	
32	23.01.04	C. Breuer	Briefpapier	Hugos Shop	20	1.100,00 €	
33	23.01.04	C. Breuer	Briefumschlä	Hugos Shop	53	7.420,00 €	
34	24.01.04	B. Weidner	Briefumschlä	Art & Design	100	10.000,00 €	
35	25.01.04	E. Sauerbier	Briefpapier	Papier & Dec	260	16.900,00 €	
36	25.01.04	E. Sauerbier	Briefumschlä	Uschi	45	5.850,00 €	
37	26.01.04	C. Breuer	Briefpapier	Papier 2002	125	10.000,00 €	
38	26.01.04	C. Breuer	Klebeetikette	Papier & Dec	30	5.250,00 €	
39	27.01.04	B. Weidner	Briefumschlä	Papier 2002	25	2.375,00 €	
40	30.01.04	E. Sauerbier	Briefumschlä	Uschi	120	12.000,00 €	
41	30.01.04	E. Sauerbier	Klebeetikette	Hugos Shop	30	6.600,00 €	
42	30.01.04	E. Sauerbier	Briefpapier	Casarossa	80	6.400,00 €	
43	31.01.04	C. Breuer	Klebeetikette	Uschi	40	8.000,00 €	
44	01.02.04	B. Weidner	Briefumschlä	Uschi	70	5.600,00 €	
45	01.02.04	B. Weidner	Klebeetikette	Uschi	20	4.600,00 €	
46	02.02.04	E. Sauerbier	Klebeetikette	Papier & Dec	70	11.900,00 €	

Abbildung 5.101 Das Ergebnis der Sortierung

Tabelle 5.78 Zum Sortieren stellt „com.sun.star.table.TableSortField“ folgende Struktur zum Sortieren für eine Spalte zur Verfügung:

Eigenschaft	Beschreibung
Field	Das Feld, nach dem sortiert wird (die Zählung beginnt bei 0!)
IsAscending	aufsteigend (oder absteigend)
IsCaseSensitive	unterscheidet nach Groß- und Kleinschreibung:

Eigenschaft	Beschreibung
	a<b<c<…<z<A<B<C<…
FieldType	Datentyp der Aufzählung com.sun.star.table.TableSortFieldType: AUTOMATIC, NUMERIC, ALPHANUMERIC

Tabelle 5.79 Die Methode zum Sortieren erhält weitere Eigenschaften

Eigenschaft	Beschreibung
SortAscending	aufsteigend (oder absteigend)
IsCaseSensitive	unterscheidet nach Groß- und Kleinschreibung: a<b<c<…<z<A<B<C<…
SortColumns	die Spalten werden bei „True" sortiert
SortFields	Array der Tabelle 5.36 – nach der sortiert wird
ContainsHeader	enthält Überschriftszeile

Es wird das Sortierkriterium in einem Datenfeld für eine Spalte festgelegt, diese wird an die Eigenschaft `SortFields` eines zweiten Datenfeldes übergeben. Will man nun nach mehreren Spalten sortieren, dann muss das erste Datenfeld erweitert werden:

```
Sub Sortieren2
   Dim objDatei as Object
   Dim objBlatt As Object
   Dim objBereich As Object
   Dim SortierKrit(3) As new com.sun.star.beans.PropertyValue
   Dim SortierFeld(1) As new com.sun.star.table.TableSortField



   objDatei = ThisComponent
   objBlatt = objDatei.Sheets(4)
   objBereich = objBlatt.getCellRangeByName("A1:C7")

   SortierFeld(0).Field = 2
   SortierFeld(0).IsAscending = True
   SortierFeld(0).FieldType = _
```

```
      com.sun.star.util.SortFieldType.ALPHANUMERIC
   SortierFeld(1).Field = 0
   SortierFeld(1).IsAscending = True
   SortierFeld(1).FieldType = _
      com.sun.star.util.SortFieldType.ALPHANUMERIC

   SortierKrit(0).Name = "SortFields"
   SortierKrit(0).Value = SortierFeld()
   SortierKrit(1).Name = "SortColumns"
   SortierKrit(1).Value = False
   SortierKrit(2).Name = "ContainsHeader"
   SortierKrit(2).Value = true
   SortierKrit(3).Name = "IsCaseSensitive"
   SortierKrit(3).Value = false

   objBereich.Sort(SortierKrit())

End Sub
```

5.9.2 Zielwertsuche

Die Methode `seekGoal` des Datei-Objekts verlangt drei Parameter: die Zelladresse der Formelzelle, die Zelladresse der „variablen Zelle“ und der Wert, auf den zurückgerechnet werden soll. Anschließend gibt das Objekt die beiden Eigenschaften Result und Divergence zurück. An einem kleinen (albernen) Beispiel soll dies gezeigt werden:

Neulich beim Roulette beschloss ich, mein ganzes Geld nur auf Rot zu setzen. Falls die Kugel auf Rot fiele, würde sich also mein Spieletat verdoppeln, und in diesem Fall sollte auch der Croupier davon profitieren: Hundert Euro Trinkgeld hielt ich für angemessen. Dann ging's los. Die erste Kugel: Rot! Mein Kapital verdoppelte sich, hundert Euro gab ich dem Croupier. Meine Glückssträhne hielt an. Auch bei der zweiten und dritten Kugel gewann Rot, mein Kapital verdoppelte sich jeweils, und jedes Mal gab ich dem Croupier hundert Euro. Bei der vierten Kugel geschah es: schon wieder Rot. Mein Spieletat wurde verdoppelt, doch als ich dem Croupier hundert Euro gab, war ich restlos pleite.

Wie hoch war der Einsatz?

Man kalkuliert die vier Runden mit einem beliebigen Wert durch, beispielsweise mit 400 €:

```
Sub Zielwertsuche
   Dim objDatei As Object
   Dim objBlatt As Object
   Dim objZelle As Object
   Dim objStartZelle As Object
   Dim objZiel As Object
   Dim intZaehler As Integer

   objDatei = ThisComponent
   If objDatei.Sheets.hasByName("Roulette") = True Then
      objBlatt = objDatei.Sheets.getByName("Roulett")
   Else
      objBlatt = objDatei.createInstance _
         ("com.sun.star.sheet.Spreadsheet")
      objDatei.Sheets.insertByName("Roulett", objBlatt)
   End If

   With objBlatt
      objZelle = .getCellByPosition(0, 0)
      objZelle.String = "Runde"
      objZelle = .getCellByPosition(1, 0)
      objZelle.String = "Anfangswert"
      objZelle = .getCellByPosition(2, 0)
      objZelle.String = "Gewinn"
      objZelle = .getCellByPosition(3, 0)
      objZelle.String = "Geschenk"

     For intZaehler = 1 To 4
         objZelle = .getCellByPosition(0, intZaehler)
         objZelle.String = "Runde 1"
```

```
        objZelle = .getCellByPosition(1, intZaehler)
        objZelle.Formula = "=D" & intZaehler
        objZelle = .getCellByPosition(2, intZaehler)
        objZelle.Formula = "=B" & intZaehler + 1 & "*2"
        objZelle = .getCellByPosition(3, intZaehler)
        objZelle.Formula = "=C" & intZaehler + 1 & "-100"
    Next

    objStartZelle = .getCellByPosition(1, 1)
    objStartZelle.Value = 400
  End With
```

Anschließend wird mit Hilfe der Zielwertsuche „zurückgerechnet“. Ebenso wie Calc kann der Wert angezeigt werden. Um ihm in die Ausgangszelle zu schreiben, muss die Eigenschaft `Result` des Objekts `seekGoal` in den `Value` der Zelle geschrieben werden. Oder er wird einfach nur angezeigt:

```
  objZiel = objDatei.seekGoal _
     (objZelle.CellAddress, objStartZelle.CellAddress, 0)
  MsgBox "Der Einsatz betrug: " & objZiel.Result & Chr(13) & _
  "Er weicht um " & objZiel.Divergence & " Euro vom gesuchten Wert ab."
End Sub
```

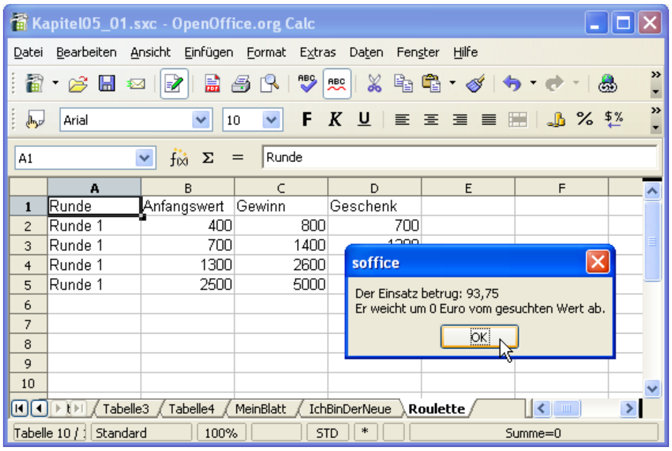


Abbildung 5.102 Die Zielwertsuche

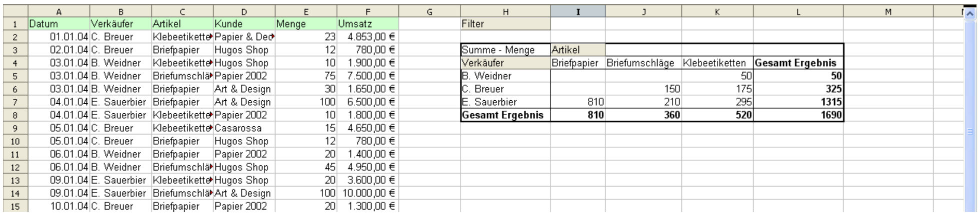
5.9.3 Der Datenpilot

Aus einer Tabelle, die eine Überschriftszeile enthält, können Daten gefiltert und zusammengefasst werden. Calc nennt diesen Assistenten „Datenpilot“.

Er kann programmiert werden:

```
Sub Datenpilot
   Dim objDatei As object
   Dim objBlatt As Object
   Dim objBereich As Object
   Dim objZiel As New com.sun.star.table.CellAddress
   Dim objDatenpilotBereich As Object
   Dim objDatenpilotTabellen As Object
   Dim objLayout As Object
   Dim objFelder As Object
   Dim objFeld As Object
   Dim i As Integer

   objDatei =    ThisComponent
   If objDatei.Sheets.hasByName("Daten") = False Then
      MsgBox "Das Blatt ""Daten"" ist nicht vorhanden."
      Exit Sub
   End If
   objBlatt = objDatei.Sheets.getByName("Daten")
   objBereich = objBlatt.getCellRangeByName("A1:F76")

   With objZiel
      .Sheet = objBereich.getRangeAddress().Sheet
      .Row = 0: .Column = 7
   End With

   objDatenpilotTabellen = objBlatt.getDataPilotTables()
   objLayout = objDatenpilotTabellen.createDataPilotDescriptor()
   objLayout.setSourceRange(objBereich.getRangeAddress())
```

```
    objFelder = objLayout.getDataPilotFields()

    objFeld = objFelder.getByIndex(1)
    objFeld.Orientation = _
       com.sun.star.sheet.DataPilotFieldOrientation.ROW

    objFeld = objFelder.getByIndex(2)
    objFeld.Orientation = _
       com.sun.star.sheet.DataPilotFieldOrientation.COLUMN

    objFeld = objFelder.getByIndex(5)
    objFeld.Orientation = _
       com.sun.star.sheet.DataPilotFieldOrientation.DATA
    objFeld.Function = com.sun.star.sheet.GeneralFunction.SUM

    If objDatenpilotTabellen.hasByName("DP") Then
       objDatenpilotTabellen.removeByName("DP")
    End If
    objDatenpilotTabellen.insertNewByName("DP", objZiel, objLayout)
    i = 7
    Do
       objBlatt.Columns(i).OptimalWidth = True
       i = i + 1
    Loop Until objBlatt.getCellByPosition(i, 3).String = ""

End Sub
```



	Datum	Verkäufer	Artikel	Kunde	Menge	Umsatz
2	01.01.04	C. Breuer	Klebeetikette	Papier & Dec	23	4.853,00 €
3	02.01.04	C. Breuer	Briefpapier	Hugos Shop	12	780,00 €
4	03.01.04	B. Weidner	Klebeetikette	Hugos Shop	10	1.900,00 €
5	03.01.04	B. Weidner	Briefumschlä	Papier 2002	75	7.500,00 €
6	03.01.04	B. Weidner	Briefpapier	Art & Design	30	1.650,00 €
7	04.01.04	E. Sauerbier	Briefpapier	Art & Design	100	6.500,00 €
8	04.01.04	E. Sauerbier	Klebeetikette	Papier 2002	10	1.800,00 €
9	05.01.04	C. Breuer	Klebeetikette	Casarossa	15	4.650,00 €
10	05.01.04	C. Breuer	Briefpapier	Hugos Shop	12	780,00 €
11	06.01.04	B. Weidner	Briefpapier	Papier 2002	20	1.400,00 €
12	06.01.04	B. Weidner	Briefumschlä	Hugos Shop	45	4.950,00 €
13	09.01.04	E. Sauerbier	Klebeetikette	Hugos Shop	20	3.600,00 €
14	09.01.04	E. Sauerbier	Briefumschlä	Art & Design	100	10.000,00 €
15	10.01.04	C. Breuer	Briefpapier	Papier 2002	20	1.300,00 €

Filter

Summe - Menge	Artikel			
Verkäufer	Briefpapier	Briefumschläge	Klebeetiketten	**Gesamt Ergebnis**
B. Weidner			50	**50**
C. Breuer		150	175	**325**
E. Sauerbier	810	210	295	**1315**
Gesamt Ergebnis	**810**	**360**	**520**	**1690**

Abbildung 5.103 Der Datenpilot

In einem Objekt, das über die Methode `createDataPilotDescriptor` erstellt wird, werden die Einstellungen des Layouts vorgenommen: das Feld beziehungsweise die Felder für die Zeilen, Spalten und Daten. Zum Löschen steht die Methode `removeByName(String)` zur Verfügung. Und auf einem Tabellenblatt wird er mit der Methode `insertNewByName(String, Zieladresse, Layout)` eingefügt.

Selbstverständlich können auch mehrere Spalten als Felder angegeben werden:

```
objFeld = objFelder.getByIndex(1)

objFeld.Orientation = com.sun.star.sheet.DataPilotFieldOrientation.ROW

objFeld = objFelder.getByIndex(3)

objFeld.Orientation = com.sun.star.sheet.DataPilotFieldOrientation.ROW
```

Oder mehrere Felder können für die Daten festgelegt werden:

```
objFeld = objFelder.getByIndex(4)

objFeld.Orientation = com.sun.star.sheet.DataPilotFieldOrientation.DATA

objFeld.Function = com.sun.star.sheet.GeneralFunction.SUM

objFeld = objFelder.getByIndex(5)

objFeld.Orientation = com.sun.star.sheet.DataPilotFieldOrientation.DATA

objFeld.Function = com.sun.star.sheet.GeneralFunction.SUM
```

Oder mehrere Funktionen für ein Datenfeld:

```
objFeld = objFelder.getByIndex(4)


objFeld.Orientation = com.sun.star.sheet.DataPilotFieldOrientation.DATA

objFeld.Function = com.sun.star.sheet.GeneralFunction.COUNT

objFeld = objFelder.getByIndex(4)

objFeld.Orientation = com.sun.star.sheet.DataPilotFieldOrientation.DATA

objFeld.Function = com.sun.star.sheet.GeneralFunction.SUM
```

Tabelle 5.80 Für die Richtung stellt der Service „com.sun.star.DataPilotFieldOrientation" folgende Aufzählung zur Verfügung:

Konstante	**Wert**	**Beschreibung**
HIDDEN	0	wird nicht angezeigt
COLUMN	1	Spalte
ROW	2	Zeile
PAGE	3	Seite
DATA	4	die Daten

Tabelle 5.81 Für die Funktionen stellt der Service „com.sun.star.GeneralFunction" folgende Aufzählung zur Verfügung:

Konstante	**Wert**	**Beschreibung**
NONE	0	keine Funktion
AUTO	1	bei Zahlenwerten wird Summe verwendet, bei alphanumerischen Inhalten Anzahl
SUM	2	Summe
COUNT	3	Anzahl
AVERAGE	4	Durchschnitt
MAX	5	Maximum
MIN	6	Minimum
PRODUCT	7	Produkt
COUNTNUMS	8	zählt nur die Zahlenwerte
STDEV	9	Standardabweichung (Stichprobe)
STDEVP	10	Standardabweichung (Grundgesamtheit)
VAR	11	Varianz (Stichprobe)
VARP	12	Varianz (Grundgesamtheit)

Über einen Filteroperator können bestimmte Werte ausgeschlossen werden.

Tabelle 5.82 Der Service „com.sun.star.sheet.FilterOperator" stellt folgende Werte zur Verfügung:

Konstante	**Beschreibung**
EMPTY	leere Zellen
NOT_EMPTY	nichtleere Zellen
EQUAL	gleich einem Wert

Konstante	Beschreibung
NOT_EQUAL	ungleich einem Wert
GREATER	>
GREATER_EQUAL	≥
LESS	<
LESS_EQUAL	≤
TOP_VALUES	Spitzenwerte
TOP_PERCENT	Spitzenwerte (nach Prozent)
BOTTOM_VALUES	niedrigste Werte
BOTTOM_PERCENT	niedrigste Werte (nach Prozent)

Sollen mehrere Bedingungen miteinander verknüpft werden, dann können sie mit den beiden logischen Konjunktoren „com.sun.star.sheet.FilterConnection.AND“ beziehungsweise „com.sun.star.sheet.FilterConnection.OR“ verbunden werden. An das DataPilotDescriptor-Objekt kann nun mit der Methode `setFilterFields` ein Datenfeld gehängt werden, das die entsprechenden Informationen beinhaltet, beispielsweise so:

```
   Dim objFilter(0) as New com.sun.star.sheet.TableFilterField
...
   objFilter(0).Field = 5
   objFilter(0).IsNumeric = True
   objFilter(0).Operator = com.sun.star.sheet.FilterOperator.LESS
   objFilter(0).NumericValue = 20000

   objLayout.FilterDescriptor.setFilterFields(objFilter())
```

Tabelle 5.83 Für den Filter stehen folgende Eigenschaften zur Verfügung:

Eigenschaft	Beschreibung
Field	die Spaltennummer des Feldes, auf das der Filter angewendet wird
Operator	einer der Werte aus der Tabelle 5.40
Connection	Verknüpfungsoperator bei mehreren Bedingungen („com.sun.star.sheet.FilterConnection.AND“ oder „com.sun.star.sheet.FilterConnection.OR“)
IsNumeric	gibt an, ob Zahlen gefiltert werden
NumericValue	Zahlenwert
StringValue	Text, der gefiltert wird

Tabelle 5.84 Und schließlich könnten noch folgende Eigenschaften an das Objekt „FilterDescriptor“ gebunden werden:

Eigenschaft	Beschreibung
IsCaseSensitive	unterscheidet nicht nach Groß- und Kleinschreibung
SkipDuplicates	übergeht Duplikate
UseRegularExpressions	interpretiert das TabelFilterField als „regular Expression“
SaveOutputPosition	speichert den Ausgabebereich
Orientation	Richtung: „com.sun.star.table.TableOrientation.ROWS“ oder „com.sun.star.table.TableOrientation.COLUMNS“
ContainsHeader	enthält Überschrift, die nicht mitgefiltert wird
CopyOutputData	kopiert an die OutputPosition
OutputPosition	Zelladresse, an die kopiert wird
MaxFieldCount	maximale Anzahl der gefilterten Felder

```
Sub Filter
   dim objDatei As object
   dim objBlatt As Object
   dim objQuellbereich As Object
   dim objFilterbereich As Object
   dim objZielbereich As Object
   dim strFilterKrit(0)
   dim strKrit As String
   dim objDesc As Object
   dim objFilter As Object 'New com.sun.star.sheet.XSheetFilterable

   objdatei =    ThisComponent
   objBlatt = objdatei.Sheets.getByName("Daten")
    objQuellbereich = objblatt.getCellRangeByName("A1:O12")
    objFilterbereich  = objBlatt.getCellRangeByName("A18:A19")
   objZielbereich = objBlatt.getCellRangeByName("A22")

   objDesc = _
      objDatei.Sheets.getByName("Daten").createFilterDescriptor(False)
```

```
   objFilter = _
      objDatei.Sheets.getByName("Daten").createFilterDescriptor(False)
   With objDesc
      .ContainsHeader = True
      .UseRegularExpressions = False
      .IsCaseSensitive = True
      .FilterFields = objFilterbereich
      .setFilterFields(objFilterbereich)
      .CopyOutputData = True
      .SaveOutputPosition = False
      .OutputPosition = objZielbereich
   End With
   objQuellbereich.Filter(objFilter)
End Sub
```

5.10 Diagramme (Charts)

Vielleicht sind Sie als versierter Programmierer der Meinung, dass man keine Diagramme programmieren muss, da das Erstellen eines Diagramms mit Hilfe des Assistenten sehr schnell geregelt ist. Dennoch gibt es eine Reihe von Fällen, in denen schnell, routinemäßig und effizient eine Reihe von Diagrammen erzeugt werden muss, oder es gibt Anwender, die vor dem Selbererstellen von Diagrammen zurückschrecken.

Auch wenn OOo Diagramme in allen Anwendungsprogrammen erstellen kann, bietet es sich an, Diagramme in Calc zu erstellen, da dort die besten Möglichkeiten existieren, Zahlen „aufzubereiten".

5.10.1 Diagramme erstellen

Diagramme werden in OOo nicht als eigenständige Dokumente behandelt, sondern als Objekte, die in ein existierendes Dokument eingebettet werden. Während Diagramme in Text- und Zeichendokumenten isoliert vom Inhalt des Dokumentes stehen, können sie in Spreadsheet-Dokumenten mit den Dokumentdaten verknüpft werden. Deshalb ist die Sammlung `Charts` eine Eigenschaft des Tabellenblattes. Die Anzahl der existierenden Diagramme wird mit der Eigenschaft `Count`

oder der Methode `getCount()` ermittelt, die Namen können in einem Array über `ElementNames` übergeben werden. Mit der Methode `hasElements()` kann überprüft werden, ob überhaupt Diagramme vorhanden sind, mit `getByName(String)` und `getByIndex(Long)` kann auf ein existierendes Diagramm zugegriffen werden. Die Methode `hasByName(String)` überprüft, ob ein bestimmtes Diagramm existiert. Während `removeByName(String)` ein Diagramm löscht, erzeugt die Methode `addNewByName` ein neues Diagramm. Sie verlangt folgende Parameter:

Tabelle 5.85 Parameter der Methode addNewByName

Name	Wert	Beschreibung
Name	String	Name des Diagramms
EmbeddedObject	Object	Diagrammfläche (die Position auf dem Tabellenblatt)
Ranges	Array	Zellbereich
HasColumnHeaders	Boolean	besitzt Spaltenköpfe
HasRowHeaders	Boolean	besitzt Zeilenköpfe

Die Diagrammfläche unterstützt den Service „com.sun.star.awt.Rectangle“ und verfügt über die vier Eigenschaften `X`, `Y`, `Width` und `Height`. Damit wird ihre Größe in 0,001 mm festgelegt. Fehlen sie, dann wird – ohne Fehlermeldung – kein Diagramm erstellt. Der Bereich mit den Eigenschaften `Sheet`, `StartColumn`, `StartRow`, `EndColumn` und `EndRow` wurde ausführlich in Kapitel 5.3 beschrieben. Damit kann nun ein Diagramm erstellt werden:

```
Sub Diagramm1
   Dim objDoc As Object
   Dim objDiagramme As Object
   Dim objDiaFlaeche As New com.sun.star.awt.Rectangle
   Dim objZellAdresse(0) As New com.sun.star.table.CellRangeAddress
   objDoc = ThisComponent
   objDiagramme = objDoc.Sheets(0).Charts

   objDiaFlaeche.X = 8000
   objDiaFlaeche.Y = 1000
   objDiaFlaeche.Width = 10000
   objDiaFlaeche.Height = 7000

   objZellAdresse(0).Sheet = 0
```

```
   objZellAdresse(0).StartColumn = 0
   objZellAdresse(0).StartRow = 0
   objZellAdresse(0).EndColumn = 1
   objZellAdresse(0).EndRow = 65

   objDiagramme.addNewByName _
      ("Diagramm1", objDiaFlaeche, objZellAdresse(), True, True)
End Sub
```

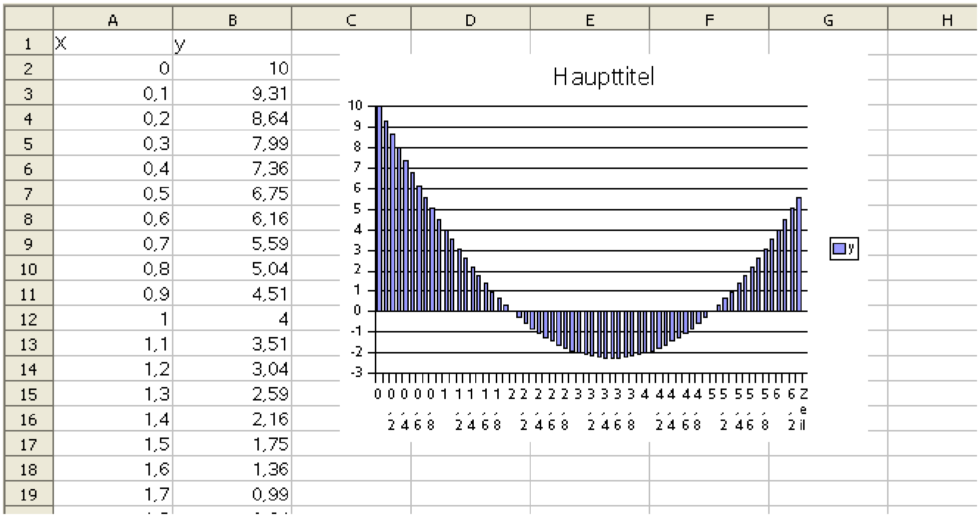


Abbildung 5.104 Ein Diagramm

Dieses Minimal-Beispiel erzeugt eine Balkengrafik. Wird eine andere Grafik benötigt, so muss das Balkendiagramm explizit ausgetauscht werden:

```
Dim objDia As Object
...
   objDia = objDiagramme.getByName("Dia1").embeddedObject
   objDia.Diagram = objDia.createInstance("com.sun.star.chart.LineDiagram")
```

5.10.2 Elemente der Diagramme

Während das Objekt `objDiagramme.getByName("Dia1")` nur die Eigenschaften `Name`, `EmbeddedObject`, `Ranges`, `HasColumnHeaders` und `HasRowHeaders` besitzt, verfügt das Objekt `objDiagramme.getByName("Dia1").embeddedObject` über folgende Eigenschaften:

Tabelle 5.86 Die Eigenschaften des Objekts „ChXChartDocument":

Eigenschaft	Wert	Beschreibung
CategoriesRangeAddress	String	enthält die Adresse der Zeilen, die für die Kategorien zuständig sind
Area	Object	der Bereich des Diagramms
Diagram	Object	der Diagrammtyp, das eigentliche Diagramm
Data	Object	die Daten
HasMainTitle	Boolean	aktiviert den Titel
HasSubTitle	Boolean	aktiviert den Untertitel
Title	Object	der Titel
SubTitle	Object	der Untertitel
HasLegend	Boolean	aktiviert die Legende
Legend	Object	die Legende

5.10.3 Objekte, die allen Diagrammen zur Verfügung stehen

Tabelle 5.87 Einige Eigenschaften des Title-, SubTitle- und Legend-Objekts:

Eigenschaft	Wert	Beschreibung
String	String	Text
CharFontName	String	Schriftart
CharWeight	Single	Schriftstärke gemäß com.sun.star.awt.FontWeight: DONTKNOW (0), THIN (50), ULTRALIGHT (60), LIGHT (75), SEMILIGHT (90), NORMAL (100), SEMIBOLD (110), BOLD (150), ULTRABOLD (175) und BLACK (200)
CharHeight	Single	Schriftgröße in Punkt (pt)
CharUnderline	Integer	Art des Unterstrichs gemäß com.sun.star.awt.FontUnderline: NONE (0), SINGLE (1), DOUBLE (2), DOTTED (3), DONTKNOW (4), DASH (5), LONGDASH (6), DASHDOT (7), DASHDOTDOT (8), SMALLWAVE (9), WAVE (10), DOUBLEWA-

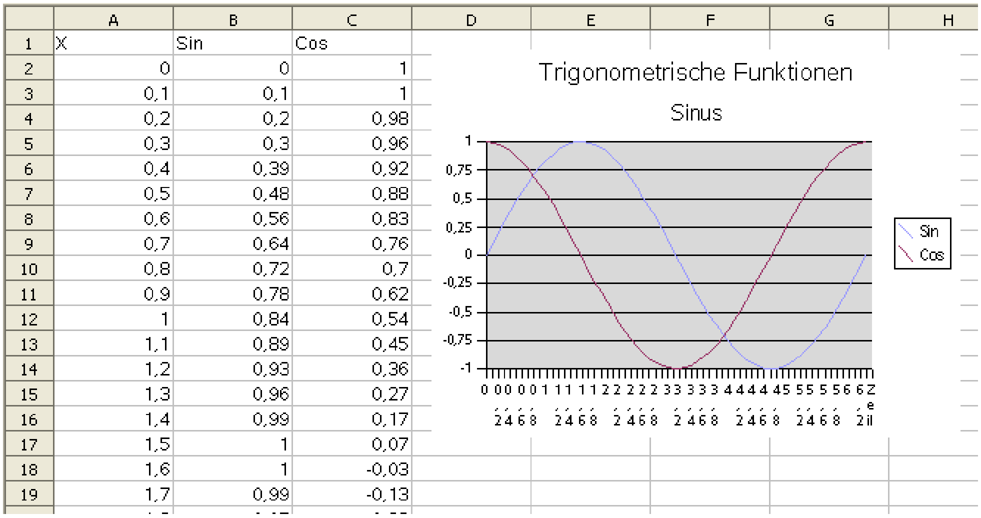
Eigenschaft	Wert	Beschreibung
		VE (11), BOLD (12), BOLDDOTTED (13), BOLDDASH (14), BOLDLONGDASH (15), BOLDDASHDOT (16), BOLDDASH-DOTDOT (17) und BOLDWAVE (18)
CharUnderlineColor	Long	Farbe des Unterstrichs
CharColor	Long	Schriftfarbe
FillColor	Long	Füllfarbe
LineColor	Long	Linienfarbe
TextRotation	Long	Drehwinkel des Textes

Die Größe und Position des Title-, SubTitle- und Legend-Objekts wird über die Objekte „Size“ und „Position“ festgelegt. Die weiteren Formatattribute der Füllfläche und Linienformate stehen in den Services „com.sun.star.drawing.FillProperties“ und „com.sun.star.drawing.LineStyle“ sowie „com.sun.star.style.CharacterProperties“ für Zeichen zur Verfügung. Sie werden in Kapitel 6 beschrieben. Die Legende kann darüber hinaus mit „Alignment“ festgelegt werden. Folgende Werte stehen über die Konstante `com.sun.star.chart.ChartLegendPosition` zur Verfügung:

BOTTOM, TOP, LEFT, RIGHT.

Ebenso besitzt jedes Diagramm eine Hintergrundfläche. Für diese Flächen ist das Objekt „Area“ verantwortlich, das den Service „com.sun.star.chart.ChartArea“ unterstützt. Er unterstützt Linien- und Flächenattribute:

```
objDia.HasMainTitle = True
objDia.Title.String = "Trigonometrische Funktionen"
objDia.HasSubTitle = True
objDia.Subtitle.String = "Sinus"
objDia.HasLegend = True
objDia.Legend.Alignment = com.sun.star.chart.ChartLegendPosition.RIGHT
objDia.Legend.FillStyle = com.sun.star.drawing.FillStyle.SOLID
objDia.Legend.FillColor = RGB(255, 255, 255)
objDia.Legend.CharHeight = 8
objDia.Area..FillColor = RGB(255, 255, 255)
```



	A	B	C
1	X	Sin	Cos
2	0	0	1
3	0,1	0,1	1
4	0,2	0,2	0,98
5	0,3	0,3	0,96
6	0,4	0,39	0,92
7	0,5	0,48	0,88
8	0,6	0,56	0,83
9	0,7	0,64	0,76
10	0,8	0,72	0,7
11	0,9	0,78	0,62
12	1	0,84	0,54
13	1,1	0,89	0,45
14	1,2	0,93	0,36
15	1,3	0,96	0,27
16	1,4	0,99	0,17
17	1,5	1	0,07
18	1,6	1	-0,03
19	1,7	0,99	-0,13

Abbildung 5.105 Ein Liniendiagramm

Während diese vier Elemente (Titel, Subtitel, Hintergrundfläche und Legende) zu jedem Diagrammtyp gehören, verfügen nur einige der Diagrammtypen über bestimmte Objekte und Eigenschaften.

5.10.4 Achsen

Diagramme verfügen über fünf verschiedene Arten von Achsen. Zweidimensionale Diagramme können neben der x- und y-Achse eine sekundäre x-, beziehungsweise y-Achse besitzen. 3D-Diagramme können eine z-Achse benutzen. Die primären x-, y- und z-Achsen können einen Titel, eine Beschreibung, ein Gitter und Hilfsgitter anzeigen.

Tabelle 5.88 Eigenschaften der primären x-, y- und z-Achse

Eigenschaft	Wert	Beschreibung
HasXAxis, HasYAxis und HasZAxis	Boolean	zeigt die Achse an oder blendet sie aus
XAxis, YAxis und ZAxis	Object	das Achsen-Objekt
HasXAxisDescription, HasYAxisDescription und HasZAxisDescription	Boolean	zeigt die Beschreibung an
HasXAxisGrid, HasYAxisGrid und HasZAxisGrid	Boolean	Hauptgitter
XMainGrid, YMainGrid und ZMainGrid	Object	Hauptinformationen

Eigenschaft	Wert	Beschreibung
HasXAxisHelpGrid, HasYAxisHelpGrid und HasZAxisHelpGrid	Boolean	Hilfsgitter
XHelpGrid, YHelpGrid und ZHelpGrid	Object	Hilfsgitterinformationen
HasXAxisTitle, HasYAxisTitle und Has-ZAxisTitle	Boolean	zeigt den Titel an
XAxisTitle, YAxisTitle und ZAxisTitle	Object	Titel

Tabelle 5.89 Für die sekundäre x- und y-Achse stehen folgende Eigenschaften zur Verfügung:

Eigenschaft	Wert	Beschreibung
HasSecondaryXAxis und HasSecondaryXAxis	Boolean	zeigt die sekundäre Achse an
SecondaryXAxis und SecondaryYAxis	Object	die sekundäre x- und y-Achse
HasSecondaryXAxisDescription und HasSecondaryXAxisDescription	Boolean	zeigt die Beschreibung an

Die Achsen-Objekte des Diagramms unterstützen den Service „com.sun.star.chart.ChartAxis“. Er stellt neben den Attributen für Zeichen (Service „com.sun.star.style.CharacterProperties“, siehe Kapitel 6) und Linien (Services „com.sun.star.drawing.LineStyle“, siehe Kapitel 8) folgende Eigenschaften zur Verfügung:

Tabelle 5.90 Die Achsen-Objekte verfügen über folgende Eigenschaften

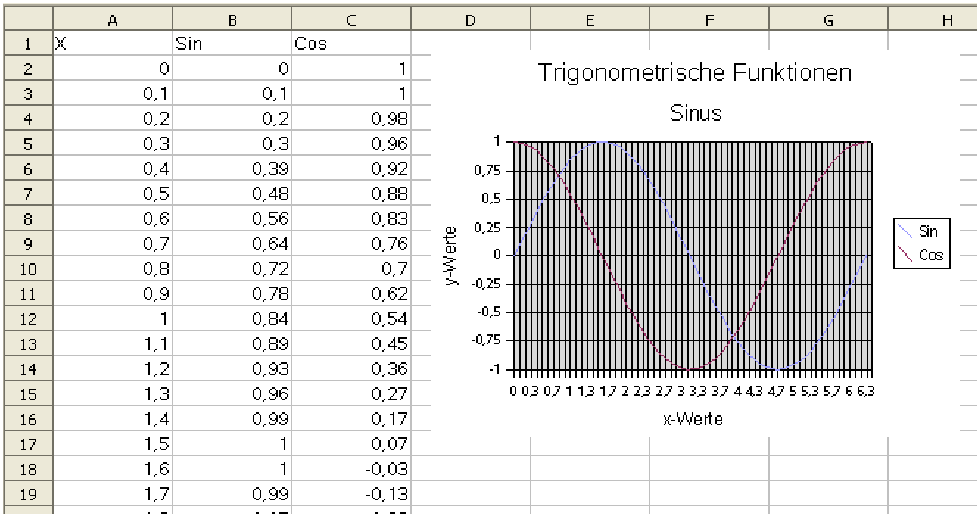
Eigenschaft	Wert	Beschreibung
Max	Double	Maximalwert für die Achse
Min	Double	Minimalwert für die Achse
Origin	Double	Schnittpunkt für kreuzende Achsen
StepMain	Double	Abstand zwischen zwei Hauptstrichen der Achse
StepHelp	Double	Abstand zwischen zwei Hilfsstrichen der Achse
AutoMax	Boolean	ermittelt automatisch den Maximalwert für die Achse
AutoMin	Boolean	ermittelt automatisch den Minimalwert für die Achse
AutoOrigin	Boolean	ermittelt automatisch den Schnittpunkt für kreuzende Achsen
AutoStepMain	Boolean	ermittelt automatisch den Abstand zwischen Hauptstrichen einer Achse
AutoStepHelp	Boolean	ermittelt automatisch den Abstand zwischen Hilfsstrichen einer Achse
Logarithmic	Boolean	skaliert die Achsen logarithmisch statt linear
DisplayLabels	Boolean	aktiviert die Textbeschriftung für Achsen

Eigenschaft	Wert	Beschreibung
TextRotation	Long	Drehwinkel der Textbeschriftung der Achsen in 100stel Grad
Marks	Const	Konstante, die angibt, ob die Hauptstriche der Achse innerhalb und/oder außerhalb der Diagrammfläche liegen sollen: Vorgabewerte gemäß com.sun.star.chart.ChartAxisMarks
HelpMarks	Const	Konstante, die angibt, ob die Hilfsstriche der Achse innerhalb und/oder außerhalb der Diagrammfläche liegen sollen: Vorgabewerte gemäß com.sun.star.chart.ChartAxisMarks
Overlap	Long	Prozentwert, der angibt, wie weit sich die Balken von verschiedenen Serien überlappen dürfen; bei 100% werden die Balken komplett übereinander dargestellt, bei 100% existiert ein Abstand von einer Balkenbreite zwischen ihnen
GapWidth	Long	Prozentwert, der angibt, welcher Abstand zwischen den verschiedenen Balkengruppen eines Charts existieren darf; bei 100 % Prozent existiert ein Abstand von einer Balkenbreite
ArrangeOrder	Enum	Angaben zur Positionierung der Beschriftung; neben der Positionierung auf einer Linie besteht die Möglichkeit, die Beschriftung alternierend auf zwei Linien zu verteilen. Vorgabewert laut com.sun.star.chart.ChartAxisArrangeOrderType
TextBreak	Boolean	erlaubt Zeilenumbrüche
TextCanOverlap	Boolean	erlaubt Textüberlappungen
NumberFormat	Long	Nummernformat. s. Kap. 5.4.6

Das Ergebnis ist nicht sehr ansprechend, aber so könnten die Achsen formatiert werden:

```
objDia.Diagram.HasXAxis = True
objDia.Diagram.HasYAxis = True
objDia.Diagram.HasXAxisDescription = True
objDia.Diagram.HasYAxisDescription = True
objDia.Diagram.XAxis.StackedText = False
objDia.Diagram.XAxis.TextBreak = False
objDia.Diagram.HasXAxisGrid = True
objDia.Diagram.HasYAxisGrid = True
objDia.Diagram.HasXAxisTitle = True
objDia.Diagram.HasYAxisTitle = True
objDia.Diagram.XAxisTitle.String = "x-Werte"
```

```
objDia.Diagram.YAxisTitle.String = "y-Werte"
```



	A	B	C
1	X	Sin	Cos
2	0	0	1
3	0,1	0,1	1
4	0,2	0,2	0,98
5	0,3	0,3	0,96
6	0,4	0,39	0,92
7	0,5	0,48	0,88
8	0,6	0,56	0,83
9	0,7	0,64	0,76
10	0,8	0,72	0,7
11	0,9	0,78	0,62
12	1	0,84	0,54
13	1,1	0,89	0,45
14	1,2	0,93	0,36
15	1,3	0,96	0,27
16	1,4	0,99	0,17
17	1,5	1	0,07
18	1,6	1	-0,03
19	1,7	0,99	-0,13

Abbildung 5.106 Das formatierte Liniendiagramm

Bei Diagrammen, deren Werte gegen +∞ oder gegen -∞ konvergieren, könnte der Bereich der y-Achse festgelegt werden:

```
objDia.Diagram.YAxis.Min = -10

objDia.Diagram.YAxis.Max = 10
```

5.10.5 Diagrammwände und Bodenfläche

Während der Diagrammhintergrund („Area") die komplette Fläche des Diagramms umfasst, beschränkt sich die Diagrammrückwand auf die Fläche unmittelbar hinter dem Datenbereich. Bei 3D-Diagrammen existieren normalerweise zwei Diagrammwände: eine hinter dem Datenbereich und eine als linke Begrenzung zur y-Achse. Zusätzlich enthalten 3D-Diagramme in der Regel eine Bodenfläche.

Tabelle 5.91 3D-Diagramme enthalten folgende wichtige Objekte

Objekt	Beschreibung
Floor	Bodenplatte des Diagramms (nur bei 3D-Diagrammen, unterstützt Service com.sun.star.chart.ChartArea).
Wall	Diagrammwände (nur bei 3D-Diagrammen, unterstützt Service com.sun.star.char-

Objekt	Beschreibung
	t.ChartArea)

Diese Objekte unterstützen den Service „com.sun.star.chart.ChartArea“, der wiederum die üblichen, bereits beschriebenen Flächen- und Linienattribute bereit hält (Services „com.sun.star.drawing.FillProperties“ und „com.sun.star.drawing.LineStyle“).

5.10.6 3D-Diagramme

Die meisten Diagramme in OOo lassen sich nicht nur als Fläche darstellen, sondern auch als 3D-Grafik. Alle Diagrammtypen, die diese Möglichkeit bieten, unterstützen den Service „com.sun.star.chart.Dim3DDiagram“. Der Service stellt lediglich die Eigenschaft `Dim3D` zur Verfügung, mit deren Hilfe die 3D-Darstellung aktiviert wird.

5.10.7 Gestapelte Diagramme

Unter einer gestapelten Diagrammdarstellung versteht man das Kumulieren der y-Werte zu einem neuen Gesamtwert. In OOo lassen sich die meisten Diagrammarten stapeln. Alle diese Diagramme unterstützen den Service „com.sun.star.chart.StackableDiagram“, der wiederum die Eigenschaften `Stacked` (die gestapelte Ansicht) und `Percent` (die prozentuale Verteilung) bereithält. Beides sind Boolean-Werte.

5.10.8 Die verschiedenen Diagrammtypen

Liniendiagramme

Liniendiagramme (Service „com.sun.star.chart.LineDiagram“) unterstützen eine -Achse, zwei y-Achsen und eine z-Achse. Die Darstellung ist als 2D- oder 3D-Grafik möglich (Service „com.sun.star.chart.Dim3DDiagram“). Die Linien sind stapelbar („com.sun.star.chart.Stackable-Diagram“).

Tabelle 5.92 Darüber hinaus stellen Liniendiagramme folgende Eigenschaften zur Verfügung:

Eigenschaft	Wert	Beschreibung
SymbolType	Konst	Symbol zur Visualisierung der Datenpunkte (Konstante gemäß `com.sun.star.chart.ChartSymbolType`)
SymbolSize	Long	Größe des Symbols zur Visualisierung der Datenpunkte in 100stel Millimetern

Eigenschaft	Wert	Beschreibung
SymbolBitmapURL	String	Dateiname der Grafik zur Visualisierung der Datenpunkte
Lines	Boolean	verbindet die Datenpunkte durch Linien
SplineType	Long	Spline-Funktion zur Glättung der Linien (0: keine Spline-Funktion, 1: kubische Splines, 2: B-Splines)
SplineOrder	Long	Polynomial-Gewicht für Splines (nur für B-Splines).
SplineResolution	Long	Anzahl der Stützpunkte für Spline-Berechnung

Flächendiagramme

Flächendiagramme (Service „com.sun.star.chart.AreaDiagram“) unterstützen eine x-Achse, zwei y-Achsen und eine z-Achse. Die Darstellung ist als 2D- oder 3D-Grafik möglich (Service „com.sun.star.chart.Dim3DDiagram“). Die Flächen sind stapelbar („com.sun.star.chart.Stackable-Diagram“).

Balkendiagramme

Balkendiagramme (Service „com.sun.star.chart.BarDiagram“) unterstützen eine x-Achse, zwei y-Achsen und eine z-Achse. Die Darstellung ist als 2D- oder 3D-Grafik möglich (Service „com.sun.star.chart.Dim3DDiagram“). Die Balken sind stapelbar („com.sun.star.chart.Stackable-Diagram“).

Tabelle 5.93 Balkendiagramme stellen folgende Eigenschaften zur Verfügung:

Eigenschaft	Wert	Beschreibung
Vertical	Boolean	stellt die Balken senkrecht oder waagerecht dar
Deep	Boolean	positioniert die Balken bei 3D-Darstellung hintereinander statt nebeneinander.
StackedBarsConnected	Boolean	verbindet die zugehörigen Balken in einem gestapelten Diagramm über Linien (nur bei horizontalen Charts verfügbar).
NumberOfLines	Long	Anzahl der Zeilen, die in einem gestapelten Diagramm als Linien statt als Balken dargestellt werden sollen.

Tortendiagramme

Tortendiagramme (Service „com.sun.star.chart.PieDiagram“) enthalten keine Achsen und sind nicht stapelbar. Die Darstellung ist als 2D- oder 3D-Grafik möglich (Service „com.sun.star.chart.-Dim3DDiagram“).

Sowohl in OOo 1.4 als auch in OOo 2.0 stürzt mir regelmäßig die Applikation ab, wenn ich per Programmierung ein Diagramm erzeuge.

Calc

Während der Zugriff auf die Komponente StarCalc lesend und schreibend erfolgt, so steht im Zentrum des Zugriffs auf StarWriter das Erzeugen von neuen Textdateien. Dabei geht es primär um die Frage, wie Text erzeugt wird, wie er formatiert wird, wie Tabellen innerhalb eines Textdokuments generiert werden und wie Zeichenobjekte, Textrahmen und andere Objekte platziert werden.

Aber nicht ausschließlich – manchmal muss auch Text überprüft oder ausgelesen werden; die Grundtechniken hierfür werden ebenfalls in diesem Kapitel beschrieben.

6 Zugriff auf Writer

Sollen nicht einfach „nur“ Texte in Dateien „zum Merken“ geschrieben werden (also datenbankähnliche Informationen gespeichert werden), sondern besteht die Aufgabe darin, per Programmierung ähnliche, sich wiederholende Dokumente zu erstellen, dann stellt der Zugriff auf den Writer und auf Textdokumente ein geeignetes Mittel dar.

6.1 Datei öffnen, speichern, drucken und schließen

Informationen zum Erzeugen, Öffnen, Speichern und Drucken von Dokumenten wurden in Kapitel 4 beschrieben, wie sie für Textdokumente (und andere Dokumentarten) herangezogen werden können.

6.2 Der Aufbau von Textdokumenten

Ein Textdokument kann im Wesentlichen vier Arten von Informationen enthalten:

- den eigentlichen Text
- Vorlagen für die Formatierung von Zeichen, Absätzen und Seiten
- nichttextuelle Elemente wie Tabellen, Grafiken und Zeichenobjekte
- globale Einstellungen für das Textdokument

6.2.1 Absätze und Absatzteile

Der Kern eines Textdokumentes besteht aus einer Folge von Absätzen. Sie sind – anders als bei Calc – weder benannt noch indiziert, so dass es keine Möglichkeit zum direkten Zugriff auf einzelne Absätze gibt. Die Absätze lassen sich jedoch mit Hilfe des in Kapitel 4 beschriebenen Enumeration-Objektes sequenziell durchlaufen. Schritt für Schritt liefert es in einer Schleife einen Absatz nach dem anderen zurück, so dass die Absätze bearbeitet werden können.

Bei der Arbeit mit dem Enumeration-Objekt ist jedoch ein Sonderfall zu beachten: Es liefert nicht nur Absätze zurück, sondern auch Tabellen (streng genommen handelt es sich in OOo Writer bei einer Tabelle um einen speziellen Absatztyp). Daher muss vor einem Zugriff auf das zurückgelieferte Objekt geprüft werden, ob es den Service „com.sun.star.text.Paragraph“ für Absätze oder den Service „com.sun.star.text.TextTable“ für Tabellen unterstützt.

Das folgende Beispiel (*Beispiele01.sxw*) durchläuft den Inhalt eines Textdokumentes in einer Schleife und teilt jeweils in einer Meldung mit, ob es sich um einen Absatz oder eine Tabelle handelt.

```
Sub Text01
   Dim objDoc As Object
   Dim objEnum As Object
   Dim objTextElement As Object
   Dim lngAbsatz As Long
   Dim lngTabelle As Long

   objDoc = ThisComponent
   objEnum = objDoc.Text.createEnumeration

   lngAbsatz = 1
   lngTabelle = 1
   Do While objEnum.hasMoreElements
      objTextElement = objEnum.nextElement
      If objTextElement.supportsService _
         ("com.sun.star.text.Paragraph") Then
         MsgBox "Der aktuelle Block enthält einen Absatz." , _
            , "Absatz Nr. " & lngAbsatz
         lngAbsatz = lngAbsatz + 1
      ElseIf objTextElement.supportsService _
         ("com.sun.star.text.TextTable") Then
         MsgBox "Der aktuelle Block enthält eine Tabelle." , _
            , "Tabelle Nr. " & lngTabelle
         lngTabelle = lngTabelle + 1
      End If
   Loop
End Sub
```

Abbildung 6.107 Der Text wird Absatz für Absatz durchlaufen.

Das Beispiel erzeugt ein Dokument-Objekt `objDoc`, das auf das aktuelle Dokument von OOo verweist. Mit Hilfe von `objDoc` wird anschließend ein Enumeration-Objekt erstellt, das die einzelnen Textteile (Absätze und Tabellen) nacheinander durchläuft und in dem Objekt `objTextElement` ablegt. Über die Methode `supportsService` prüft das Beispiel, ob es sich bei `objTextElement` um einen Absatz oder eine Tabelle handelt, und gibt dann eine entsprechende Meldung aus.

Die Eigenschaft `objTextElement.String` liest den Inhalt des Absatzes aus. Ist der Absatz leer, das heißt, hat der Benutzer zwei Mal <Enter> gedrückt, dann ist die Länge des Strings = 0, was mit der Funktion `Len` überprüft werden kann.

Analog zu Texten und Tabellen kann ein Dokument auch nach anderen Elementen, die keinen Namen haben, durchlaufen werden. Das nächste Beispiel (*Beispiele04.sxw/Standard/ZeichenFormatieren*) zeigt, wie sämtliche Textfelder eines Textdokumentes in einer Schleife durchlaufen und auf ihren jeweiligen Typ hin überprüft werden können.

```
Sub AlleFelder
   Dim objDoc As Object
   Dim objTextFeldEnum As Object
   Dim objTextFeld As Object
   Dim strFelder As String
   objDoc = ThisComponent
   objTextFeldEnum = objDoc.getTextFields.createEnumeration
   While objTextFeldEnum.hasMoreElements()
      objTextFeld = objTextFeldEnum.nextElement()
      If objTextFeld.supportsService _
         ("com.sun.star.text.TextField.DateTime") Then
         strFelder = strFelder & Chr(13) & "Datum/Uhrzeit"
```

```
      ElseIf objTextFeld.supportsService _
         ("com.sun.star.text.TextField.Annotation") Then
         strFelder = strFelder & Chr(13) & "Notiz"
      ElseIf objTextFeld.supportsService _
         ("com.sun.star.text.TextField.FileName") Then
         strFelder = strFelder & Chr(13) & "Dateiname"
      Else
         strFelder = strFelder & Chr(13) & "ein anderes Feld"
      End If
   Wend
   MsgBox strFelder
End Sub
```

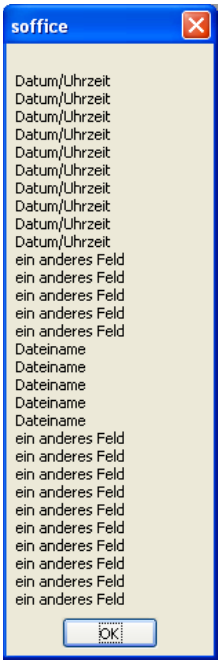


Abbildung 6.108 Die Felder werden ausgelesen.

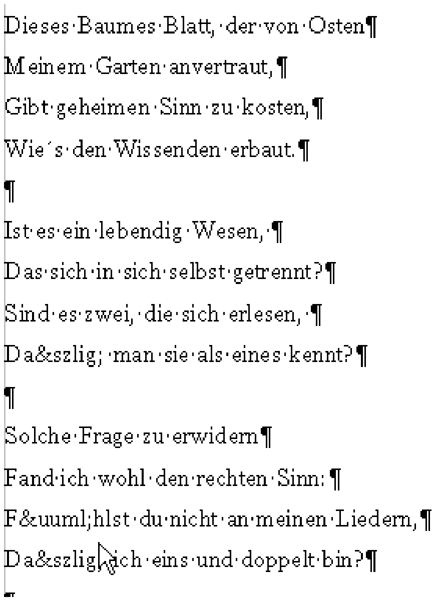
Die vollständige Liste aller Felder finden Sie in Kapitel 6.17.

6.2.2 Absätze

Der Service „com.sun.star.text.Paragraph“ gestattet Zugriff auf den Inhalt eines Absatzes. Der innerhalb des Absatzes vorhandene Text kann über die Eigenschaft `String` ausgelesen und geändert werden:

```
Sub Text02
   Dim objDoc As Object
   Dim objEnum As Object
   Dim objTextElement As Object

   objDoc = ThisComponent
   objEnum = objDoc.Text.createEnumeration

   Do While objEnum.hasMoreElements
      objTextElement = objEnum.nextElement
      If objTextElement.supportsService _
         ("com.sun.star.text.Paragraph") Then
         objTextElement.String = _
            Replace(objTextElement.String, "Ä", "&Auml;")
         objTextElement.String = _
            Replace(objTextElement.String, "Ö", "&Ouml;")
         objTextElement.String = _
            Replace(objTextElement.String, "Ü", "&Uuml;")
         objTextElement.String = _
            Replace(objTextElement.String, "ä", "&auml;")
         objTextElement.String = _
            Replace(objTextElement.String, "ö", "&ouml;")
         objTextElement.String = _
            Replace(objTextElement.String, "ü", "&uuml;")
         objTextElement.String = _
            Replace(objTextElement.String, "ß", "&szlig;")
      End If
   Loop
```

```
End Sub

Function Replace(strText As String, strZeichen1 As String, _
   strZeichen2 As String)
      Dim strTeile() As String
      Dim i As Integer

      If Instr(1, strText, strZeichen1) > 0 Then
         strTeile = Split(strText, strZeichen1)
         For i = LBound(strTeile) To UBound(strTeile)
            Replace = Join(strTeile, strZeichen2)
         Next i
      Else
         Replace = strText
      End If

End Function
```



Dieses·Baumes·Blatt,·der·von·Osten¶
Meinem·Garten·anvertraut,¶
Gibt·geheimen·Sinn·zu·kosten,¶
Wie´s·den·Wissenden·erbaut.¶
¶
Ist·es·ein·lebendig·Wesen,·¶
Das·sich·in·sich·selbst·getrennt?¶
Sind·es·zwei,·die·sich·erlesen,·¶
Daß·man·sie·als·eines·kennt?¶
¶
Solche·Frage·zu·erwidern¶
Fand·ich·wohl·den·rechten·Sinn:¶
Fühlst·du·nicht·an·meinen·Liedern,¶
Daßich·eins·und·doppelt·bin?¶

Abbildung 6.109 Text wird ersetzt.

Das Beispiel (*Beispiele01.sxw*) öffnet das aktuelle Textdokument und durchwandert dieses unter Zuhilfenahme des Enumeration-Objektes. In sämtlichen Absätzen greift es über die Eigenschaft `TextElement.String` auf die jeweiligen Absätze zu und ersetzt die Zeichenfolgen „Ä", „Ö", „Ü" …"ß" durch „Auml;", „Ouml", „Uuml" … „szlig". Die Funktion wird dabei aufgerufen und verwendet.

Wenn Sie sicher sind, dass ein Dokument nicht aus Tabellen und anderen nicht-textuellen Elementen besteht, dann können Sie mit der Methode `getString()` auf den Inhalt zugreifen:

```
Sub Text03
   MsgBox ThisComponent.Text.getString()
End Sub
```

Mit der Methode `setString(String)` kann umgekehrt Text geschrieben werden (*Beispiel01.sxw*):

```
Sub Text04
   Dim objDoc As Object
   Dim docArgs() As New com.sun.star.beans.PropertyValue
   Dim strUrl As String
```

```
    Const T As String = "An den Ufern der Havel lebte, um die Mitte " & _
    "des sechzehnten Jahrhunderts, ein Roßhändler, namens Michael " & _
    "Kohlhaas, Sohn eines Schulmeisters, einer der rechtschaffensten " & _
    "zugleich und entsetzlichsten Menschen seiner Zeit."
    strUrl = "private:factory/swriter"
    objDoc = StarDesktop.loadComponentFromURL _
       (strUrl, "_blank", 0, docArgs())

    objDoc.Text.setString(T)

End Sub
```

Die Schnittstelle „XTextRange“ stellt sicherlich das wichtigste Objekt in Writer zur Verfügung. Ihre Aufgabe ist es, den Anfang und das Ende eines Textobjektes zu bestimmen, das Textobjekt selbst festzulegen und Text hineinzuschreiben beziehungsweise auszulesen.

Tabelle 6.94 com.sun.star.text.XTextRange stellt folgende fünf Methoden bereit:

Methode	Beschreibung
getText()	gibt das Textrange-Objekt zurück
getStart()	gibt die Startposition zurück
getEnd()	gibt die Endposition zurück
setString(String)	schreibt Text zwischen Start- und Endposition
getString()	liefert die Zeichenkette

Normalerweise werden Texte nicht per Programmierung vollständig erzeugt, sondern in vorhandene Vorlagen werden an spezifischen Sprungmarken (Textmarken) Texte geschrieben. Wie dies funktioniert, wird weiter unten (Kapitel 6.15) erläutert.

6.3 Texte erzeugen

6.3.1 Das Cursor-Objekt

Um in ein leeres Dokument Texte zu schreiben, können Sie das Cursor-Objekt verwenden. Es wird mit der Methode `createTextCursor()` erzeugt und stellt flexiblere Methoden zur Verfügung als `Text.setString()`. Mit seiner Hilfe kann man sich durch ein Dokument bewegen und Texte erzeugen.

```
Dim objDoc As Object
Dim objCursor As Object
objDoc = ThisComponent
objCursor = objDoc.Text.createTextCursor()
```

`objCursor` setzt den Cursor an den Beginn des leeren Dokumentes. Dort kann nun ein Text eingefügt werden:

```
objCursor.String = "§ 1 Abfindung"
```

Wenn Sie einen Tabulator als Trennzeichen zwischen § 1 und dem Text „Abfindung“ eingeben möchten, dann können Sie den Tabulator in die Zeichenkette einfügen:

```
objCursor.String = "§ 1     Abfindung"
```

Oder Sie verketten das ASCII-Zeichen 9 (Tab):

```
objCursor.String = "§ 1" & Chr(9) & "Abfindung"
```

Selbstverständlich könnte das gesamte Dokument mit dem Befehl

```
ThisComponent.Text.setString("")
```

gelöscht werden.

Recht flexibel ist auch die Methode `createTextCursorByRange(XTextRange)`, die einen Textcursor nicht am Anfang des Dokumentes, sondern an einer bestimmten Stelle (beispielsweise an einer Textmarkenposition) erstellt. Darauf wird weiter unten (Kapitel 6.15) eingegangen:

```
objDatei = ThisComponent
objTextmarken = objDatei.getBookmarks()
If objTextmarken.hasByName("hier") = True Then
   objTextmarke = objTextmarken.getByName("hier")
   objCursor = _
      objDatei.Text.createTextCursorByRange(objTextmarke.Anchor)
   objCursor.String = "Vorgesetztenfunktion mit " & _
```

```
        "Übertragung besonderer Unternehmerpflichten"
   End If
```

Um einen Absatz einzufügen, kann leider nicht `Chr(13)` oder `Chr(10)` hinzuverkettet werden. Das Textdokument stellt die Methode `insertControlCharacter` zur Verfügung:

```
   objDoc.Text.insertControlCharacter(objCursor, 0, False)
```

Der Wert 0 steht dabei für Absatz. Er kann in einer Konstanten ausgelagert werden. Der Parameter `False` im Aufruf der Methode `insertControlCharacter` sorgt dafür, dass der aktuell vom Textcursor markierte Bereich durch die Einfügeoperation bestehen bleibt. Wird hier der Parameter „True" übergeben, so ersetzt `insertControlCharacter` den aktuellen Text.

Das folgende Beispiel (*Beispiele02.sxw*) erstellt die ersten beiden Absätze eines Vertrages, wobei zwei Geldbeträge dynamisch eingefügt werden:

```
Sub Schreiben01


   Dim objDoc As Object
   Dim objCursor As Object
   Dim sngBetrag1 As Single
   Dim sngBetrag2 As Single
   Dim docArgs() As New com.sun.star.beans.PropertyValue
   Dim strUrl As String
   sngBetrag1 = InputBox("Bitte Betrag der Abfindung eingeben!")
   sngBetrag2 = InputBox("Bitte Betrag der Höhe der Strafe " & _
      "bei Zuwiderhandlung eingeben!")

   Const ABSATZ As Integer = 0
   Const T1 As String = "Der Mitarbeiter erhält anlässlich der " & _
      "Beendigung des Anstellungsverhältnisses wegen Verlustes " & _
      "des Arbeitsplatzes eine Abfindung in Höhe von brutto € "
   Const T2 As String = ". Diese Abfindung wird ohne Abzug " & _
      "von Sozialversicherungsabgaben und in den Grenzen " & _
      "des § 3 Nr. 9 EStG einkommenssteuerfrei ausgezahlt. " & _
      "Die Abfindung wird fällig am letzten Tag des " & _
      "Arbeitsverhältnisses."
   Const T3 As String = "Voraussetzung für die Auszahlung ist, " & _
```

```
    "dass das Anstellungsverhältnis aufgrund dieses " & _
    "Aufhebungsvertrages zum oben genannten Zeitpunkt endet."
Const T4 As String = "Der Mitarbeiter verpflichtet sich, " & _
    "die Höhe der Abfindung jedermann gegenüber geheimzuhalten, " & _
    "soweit nicht eine gesetzliche Regelung entgegensteht. " & _
    "Bei Zuwiderhandlungen gegen diese Geheimhaltungspflicht " & _
    "ist zugunsten des Unternehmens, und zwar für jeden " & _
    "einzelnen Zuwiderhandlungsfall, eine Vertragsstrafe in " & _
    "Höhe von € "
Const T5 As String = " erwirkt."

strUrl = "private:factory/swriter"
objDoc = StarDesktop.loadComponentFromURL _
    (strUrl, "_blank", 0, docArgs())

objCursor = objDoc.Text.createTextCursor()
objCursor.String = "§ 1 Abfindung"
objCursor.goRight(13, False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = "1." & Chr(9)
objCursor.goRight(4, False)
objCursor.String = T1 & Format(sngBetrag1, "#,##0.00")  & T2
objCursor.goRight(len(T1 & Format(sngBetrag1, "#,##0.00")  & T2), _
    False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.goRight(1, False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = T3
objCursor.goRight(len(T3), False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.goRight(1, False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
```

```
    objCursor.String = "2." & Chr(9)
    objCursor.goRight(4, False)
    objCursor.String = T4 & Format(sngBetrag2, "#,##0.00") & T5

End Sub
```

§·1·Abfindung¶
1. → Der·Mitarbeiter·erhält·anlässlich·der·Beendigung·des·Anstellungsverhältnisses·wegen Verlustes·des·Arbeitsplatzes·eine·Abfindung·in·Höhe·von·brutto·€·1.250,00·.Diese·Abfindung·wird ohne·Abzug·von·Sozialversicherungsabgaben·und·in·den·Grenzen·des·§·3·Nr.·9·EStG einkommenssteuerfrei·ausgezahlt.·Die·Abfindung·wird·fällig·am·letzten·Tag·des Arbeitsverhältnisses.¶
¶
Voraussetzung·für·die·Auszahlung·ist,·dass·das·Anstellungsverhältnis·aufgrund·dieses Aufhebungsvertrages·zum·oben·genannten·Zeitpunkt·endet.¶
¶
2. → Der·Mitarbeiter·verpflichtet·sich,·die·Höhe·der·Abfindung·jedermann·gegenüber geheimzuhalten,·soweit·nicht·eine·gesetzliche·Regelung·entgegensteht.·Bei·Zuwiderhandlungen gegen·diese·Geheimhaltungspflicht·ist·zugunsten·des·Unternehmens,·und·zwar·für·jeden·einzelnen Zuwiderhandlungsfall,·eine·Vertragsstrafe·in·Höhe·von·€·5.000,00·verwirkt.¶

Abbildung 6.110 Der per Programmierung erstellte Vertrag

Neben dem Absatz bietet OOo weitere Steuerzeichen an, die in den Text eingefügt werden können. Die Steuerzeichen, die für Texte benötigt werden, sind in der Konstantengruppe „com.sun.star.text.ControlCharacter" definiert.

Tabelle 6.95 Folgende Steuerzeichen sind in OOo verfügbar:

Konstante	Wert	Bedeutung
PARAGRAPH_BREAK	0	Absatzumbruch
LINE_BREAK	1	Zeilenumbruch innerhalb eines Absatzes
SOFT_HYPHEN	3	mögliche Stelle für eine Silbentrennung (bedingter Trennstrich)
HARD_HYPHEN	2	obligatorische Stelle für eine Silbentrennung (erzwungener Trennstrich)
HARD_SPACE	4	geschütztes Leerzeichen, das auch im Blocksatz nicht gespreizt oder gepresst wird
APPEND_PARAGRAPH	5	hängt einen neuen Absatz an

Ein Absatz kann also auch folgendermaßen eingefügt werden, was allerdings nicht gerade die Lesbarkeit erhöht:

```
objDoc.Text.insertControlCharacter(objCursor, _
    com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, False)
```

6.4 Durch ein Dokument navigieren

Tabelle 6.96 Das Cursorobjekt stellt folgende Methoden zur Verfügung:

Methode	**Bedeutung**
goLeft (Count, Expand)	bewegt den Cursor Count Zeichen nach links
goRight (Count, Expand)	bewegt den Cursor Count Zeichen nach rechts
gotoStart (Expand)	bewegt den Cursor zum Anfang des Dokumentes
gotoEnd (Expand)	bewegt den Cursor zum Ende des Dokumentes
gotoRange (SbxOBJECT, Expand)	bewegt den Cursor zum angegebene Textrange-Objekt
gotoNextSentence (Expand)	bewegt den Cursor zum nächsten Satz
gotoPreviousSentence (Expand)	bewegt den Cursor zum vorhergehenden Satz
gotoStartOfSentence (Expand)	bewegt den Cursor zum Anfang des Satzes
gotoEndOfSentence (Expand)	bewegt den Cursor zum Ende des Satzes
gotoNextWord (Expand)	bewegt den Cursor zum nächsten Wort
gotoPreviousWord (Expand)	bewegt den Cursor zum vorhergehenden Wort
gotoStartOfWord (Expand)	bewegt den Cursor zum Anfang des Wortes
gotoEndOfWord (Expand)	bewegt den Cursor zum Ende des Wortes
gotoNextParagraph (Expand)	bewegt den Cursor zum nächsten Absatz
gotoPreviousParagraph (Expand)	bewegt den Cursor zum vorhergehenden Absatz
gotoStartOfParagraph (Expand)	bewegt den Cursor zum Anfang des Absatzes
gotoEndOfParagraph (Expand)	bewegt den Cursor zum Ende des Absatzes

Ein Satz in OOo ist die Zeichenkette bis zum nächsten Punkt oder Absatzzeichen. Deshalb werden auch Punkte hinter Abkürzungen („z.B.“) als Satzendezeichen interpretiert.

Der boolesche Parameter `Expand` gibt an, ob der Cursor ausgedehnt wird oder nicht. Das obere Beispiel könnte also auch beispielsweise wie folgt erzeugt werden:

```
objDoc = ThisComponent
objCursor = objDoc.Text.createTextCursor()
objCursor.String = "§ 1 Abfindung"
objCursor.gotoEnd(False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = "1." & Chr(9)
objCursor.gotoEndOfParagraph(False)
objCursor.String = T1 & Format(sngBetrag1, "#,##0.00")  & T2
objCursor.gotoEndOfParagraph(False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.gotoNextParagraph(False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = T3
objCursor.gotoEndOfSentence(False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.gotoNextParagraph(False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = "2." & Chr(9)
objCursor.gotoEndOfParagraph(False)
objCursor.String = T4 & Format(sngBetrag2, "#,##0.00") & T5

End Sub
```

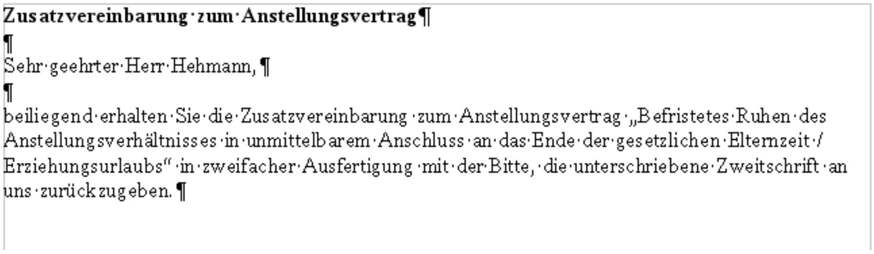
Der Vollständigkeit halber soll der Befehl `insertTextContent(TextRange, TextContent, boolean)` erwähnt werden. Mit seiner Hilfe kann nicht nur Text eingefügt werden, sondern andere Textelemente, wie: Tabellen, Textrahmen, Textmarken, Textfelder und so weiter. Gelöscht werden sie mit `removeTextContent(TextContent)`. Diese beiden Befehle werden in Kapitel 6.14–6.17 beschrieben.

6.5 Cursor ausdehnen

Soll nun ein Textteil formatiert werden, dann kann der Cursor ausgedehnt werden (*Beispiele02.sxw*):

```
Sub Schreiben03

   Dim objDoc As Object
   Dim objCursor As Object
   Dim strAnrede As Single

   strAnrede = InputBox("Bitte Anrede eingeben: " & _
      Chr(13) & """Frau xy"" oder ""Herr xy""")
   If InStr(1, strAnrede, "Frau") > 0 Then
      strAnrede = "Sehr geehrte " & Trim(strAnrede) & ","
   Else
      strAnrede = "Sehr geehrter " & Trim(strAnrede) & ","
   End If

   Const ABSATZ As Integer = 0
   Const T1 As String = "beiliegend erhalten Sie die " & _
      "Zusatzvereinbarung zum Anstellungsvertrag " & _
      "„Befristetes Ruhen des Anstellungsverhältnisses " & _
      "in unmittelbarem Anschluss an das Ende der " & _
      "gesetzlichen Elternzeit/Erziehungsurlaubs‟ in " & _
      "zweifacher Ausfertigung mit der Bitte, " & _
      "die unterschriebene Zweitschrift an uns zurückzugeben."

   objDoc = ThisComponent
   objCursor = objDoc.Text.createTextCursor()
   objCursor.String = "Zusatzvereinbarung zum Anstellungsvertrag"

   objCursor.gotoStart(False)
```

```
    objCursor.gotoEnd(True)
    objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD

    objCursor.gotoEnd(False)
    objCursor.CharWeight = com.sun.star.awt.FontWeight.NORMAL
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objCursor.String = strAnrede
    objCursor.gotoEnd(False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objCursor.String = T1

End Sub
```



Zusatzvereinbarung·zum·Anstellungsvertrag¶
¶
Sehr·geehrter·Herr·Hehmann,¶
¶
beiliegend·erhalten·Sie·die·Zusatzvereinbarung·zum·Anstellungsvertrag·„Befristetes·Ruhen·des Anstellungsverhältnisses·in·unmittelbarem·Anschluss·an·das·Ende·der·gesetzlichen·Elternzeit·/ Erziehungsurlaubs“·in·zweifacher·Ausfertigung·mit·der·Bitte,·die·unterschriebene·Zweitschrift·an uns·zurückzugeben.¶

Abbildung 6.111 Text wird erzeugt und formatiert.

Der Befehl

```
    objCursor.gotoStart(False)
```

setzt den Cursor an den Anfang des Dokumentes. Von dort aus wird eine Markierung bis zum Ende des Dokumentes eingeschaltet und diese Markierung formatiert:

```
    objCursor.gotoEnd(True)
    objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
```

Anschließend muss der Fettdruck wieder ausgeschaltet werden. Es würde auch so funktionieren:

```
    objDoc = ThisComponent
    objCursor = objDoc.Text.createTextCursor()
```

```
   objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
   objCursor.String = "Zusatzvereinbarung zum Anstellungsvertrag"


   objCursor.gotoEnd(False)
   objCursor.CharWeight = com.sun.star.awt.FontWeight.NORMAL
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
[...]
```

Ein Textcursor in der OOo API ist vergleichbar mit dem sichtbaren Cursor in einem OOo-Dokument. Er markiert eine bestimmte Stelle innerhalb eines Textdokumentes und lässt sich per Befehl in verschiedene Richtungen navigieren. Dennoch dürfen Sie die in OOo Basic verfügbaren Textcursor-Objekte nicht mit dem sichtbaren Cursor verwechseln. Es handelt sich um zwei verschiedene Dinge.

Die Textcursor-Objekte in OOo Basic agieren unabhängig von dem sichtbaren Cursor in einem Textdokument. Eine programmgesteuerte Positionsänderung eines Textcursor-Objektes hat keinerlei Auswirkungen auf den sichtbaren Cursor. Es ist sogar möglich, für ein und dasselbe Dokument mehrere Textcursor-Objekte zu öffnen, die unabhängig voneinander in verschiedenen Positionen eingesetzt werden.

Die Ausdehnung des Cursors wird dann benötigt, wenn innerhalb einer existierenden Zeichenkette ein Teil formatiert werden soll:

```
   objCursor.String = T1
   For i = 1 To 4
      ObjCursor.GoToNextWord(False)
   Next i
   objCursor.GoToEndOfWord(True)
   objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
   End Sub
```

Jede der GoTo-Anweisungen kann einen booleschen Wert zurückgeben, mit dem überprüft werden kann, ob ein weiterer Absatz, ein weiterer Satz oder ein weiteres Wort existiert, das heißt, ob die Navigation erfolgreich war oder ob die Aktion mangels Text abgebrochen wurde. Das nächste Beispiel (*Beispiel02.sxw*) durchläuft alle Absätze und meldet den Inhalt jedes Absatzes:

```
Sub AlleAbsaetze
   Dim objDoc As Object
```

```
    Dim objCursor As Object
    Dim blnFortschritt As Boolean
    Dim i As Integer

    objDoc = ThisComponent
    objCursor = objDoc.Text.createTextCursor
    i = 1

    Do
       objCursor.gotoEndOfParagraph(True)
       MsgBox objCursor.String, , "Absatz Nr. " & i
       blnFortschritt = objCursor.gotoNextParagraph(False)
       i = i +1
    Loop While blnFortschritt

End Sub
```

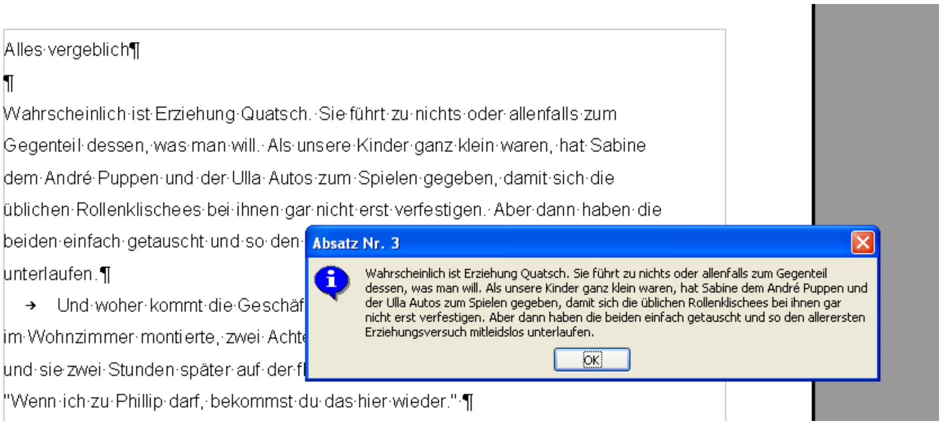


Abbildung 6.112 Die Absätze werden durchlaufen und ausgelesen.

Einige weitere Methoden sind in diesem Zusammenhang wichtig:

1. **Methode**	1. **Rück-gabe**	1. **Beschreibung**

	wert	
2. isStartOfParagraph()	2. Boolean	2. liefert True, falls der Textcursor am Anfang eines Absatzes steht.
3. isEndOfParagraph()	3. Boolean	3. liefert True, falls der Textcursor am Ende eines Absatzes steht.
4. isStartOfWord()	4. Boolean	4. liefert True, falls der Textcursor am Anfang eines Wortes steht.
5. isEndOfWord()	5. Boolean	5. liefert True, falls der Textcursor am Ende eines Wortes steht.
6. collapseToStart()	6. Void	6. setzt die Markierung zurück und positioniert den Textcursor am Beginn des bisher markierten Bereiches.
7. collapseToEnd()	7. Void	7. setzt die Markierung zurück und positioniert den Textcursor am Ende des bisher markierten Bereiches.
8. isCollapsed()	8. Boolean	8. gibt True aus, falls der Textcursor aktuell keine Markierung umfasst.
9. getString()	9. String	9. liefert die Zeichenkette des Cursors.
10. setString(String)	10. Void	10. ändert die Zeichenkette.

6.6 Der sichtbare Cursor

Manchmal ist es nötig, den sichtbaren Cursor an eine bestimmte Stelle im Dokument zu bewegen, beispielsweise wenn ein Dokument per Programmierung erstellt wurde und der Benutzer danach eine bestimmte Textstelle angezeigt bekommt, um ihm das Weiterarbeiten zu erleichtern. Für diesen Zweck stellt OOo das Objekt `getViewCursor()` zur Verfügung.

Tabelle 6.97 Der sichtbare Cursor verfügt über folgende Methoden:

Methode	Beschreibung
getText	gibt den (markierten) Text als Objekt zurück
getString	gibt den (markierten) Text als Zeichenkette zurück
getStart	gibt den Anfang (des markierten Textes) zurück
getEnd	gibt das Ende (des markierten Textes) zurück
setString(String)	schreibt Text
collapseToStart	dehnt bis zum Anfang aus
collapseToEnd	dehnt bis zum Ende aus

Methode	Beschreibung
goLeft(n, Boolean) goRight(n, Boolean) goDown(n, Boolean) goUp (n, Boolean)	bewegt den Cursor um n (ein Integer-Wert) Zeichen nach links oder rechts, um n Zeilen nach unten oder oben
gotoEndOfLine (Boolean) gotoStartOfLine (Boolean)	gehe zum Ende oder Anfang der Zeile
gotoStart(Boolean) gotoEnd(Boolean)	gehe zum Anfang oder Ende des Dokumentes
gotoRange (Object, Boolean)	gehe zu einer bestimmten Textstelle
jumpToFirstPage() jumpToLastPage() jumpToNextPage() jumpToPreviousPage() jumpToPage(n)	gehe zur ersten, letzten, nächsten, vorhergehenden Seite oder zur Seite Nummer n
jumpToEndOfPage() jumpToStartOfPage	gehe zum Ende oder zum Anfang der Seite
screenDown() screenUp()	blättere eine „Seite“ nach unten oder oben
isCollapsed()	ist der Cursor „ausgedehnt“?
isVisible()	ist der Cursor „sichtbar“?
isAtStartOfLine() isAtEndOfLine()	ist der Cursor am Anfang oder Ende einer Zeile?
setVisible(Boolean)	mache den Cursor sichtbar oder unsichtbar
getPosition()	liefert die Position (eine Struktur gemäß com.sun.star.awt.Point)
getPage()	liefert die Seitennummer (Integer)

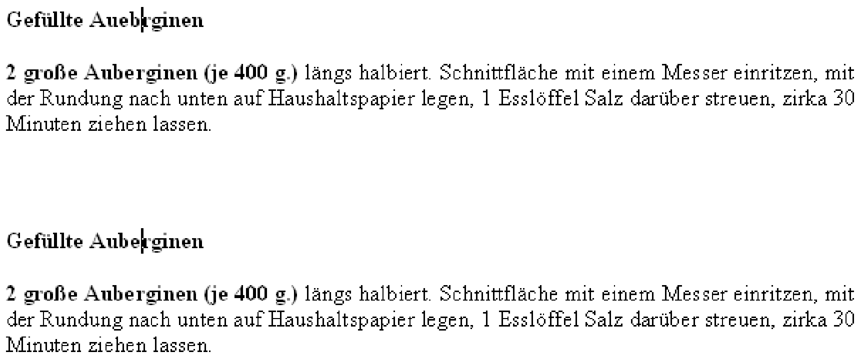
Im folgenden Beispiel wird der Cursor an den Anfang des Dokumentes gesetzt:

```
ThisComponent.CurrentController.getViewCursor().goToStart(False)
```

Ist der Text länger als eine Bildschirmseite, dann bleibt die aktuelle Bildschirmposition stehen, der Cursor „verschwindet“ nach oben.

Jedoch könnte man mit dieser Methode einige Lösungen programmieren. Das folgende Beispiel (*Beispiele04.sxw*) vertauscht die beiden Zeichen vor dem Cursor – interessant für Leute, die sehr häufig Buchstabendreher erzeugen:

```
Sub Buchstabendreher
   Dim strZeichen As String

   With StarDesktop.CurrentComponent.CurrentController.getViewCursor()
      .goLeft(1, True)
      strZeichen = .getString
      .setString("")
      .goLeft(1, False)
      .setString(strZeichen)
      .collapseToStart()
      .goRight(2, False)
   End With

End Sub
```


Gefüllte Auebrginen

2 große Auberginen (je 400 g.) längs halbiert. Schnittfläche mit einem Messer einritzen, mit der Rundung nach unten auf Haushaltspapier legen, 1 Esslöffel Salz darüber streuen, zirka 30 Minuten ziehen lassen.

Gefüllte Auberginen

2 große Auberginen (je 400 g.) längs halbiert. Schnittfläche mit einem Messer einritzen, mit der Rundung nach unten auf Haushaltspapier legen, 1 Esslöffel Salz darüber streuen, zirka 30 Minuten ziehen lassen.

Abbildung 6.113 Der Buchstabendreher wird beseitigt.

Beachten Sie, dass die Methode `setString` den Cursor ausdehnt. Wenn Sie ihn nicht per Programmierung „ausschalten“ (`.collapseToStart()`), dann würde der Cursor bei

weiteren Cursorbewegungen mit der Pfeiltaste sich „ausdehnen“. Erst die Taste <esc> unterbricht diesen „Ausdehnungsmodus“.

Zwar könnte man sich ein Symbol einfügen, mit dem die Autotexte aufgerufen werden (es funktioniert auch mit der Tastenkombination <Strg>+<F3>), zwar kann ein in den Text geschriebenes Kürzel mit der Funktionstaste <F3> durch den dahinter liegenden Autotext ersetzt werden, aber problematisch wird es, wenn Sie ein Sonderzeichen, beispielsweise aus einer anderen Sprache, häufig im Text benötigen und nicht jedes Mal die Tabelle der Sonderzeichen aufrufen möchten. Dann hilft ein kleines Makro (hier acht Makros für sämtliche türkischen Buchstaben, die nicht im deutschen Alphabeth vorkommen (*Beispiele04.sxw*):

```
Sub Klein_tsche
   StarDesktop.CurrentComponent.CurrentController.getViewCursor()._
      setString(Chr(0231))
   StarDesktop.CurrentComponent.CurrentController.getViewCursor()._
      collapseToEnd
End Sub

Sub Gross_Tsche
   StarDesktop.CurrentComponent.CurrentController.getViewCursor()._
      setString(Chr(0199))
   StarDesktop.CurrentComponent.CurrentController.getViewCursor()._
      collapseToEnd
End Sub

Sub Klein_sche
   StarDesktop.CurrentComponent.CurrentController.getViewCursor()._
      setString(Chr(351))
   StarDesktop.CurrentComponent.CurrentController.getViewCursor()._
      collapseToEnd
End Sub

Sub Gross_Sche
   StarDesktop.CurrentComponent.CurrentController.getViewCursor()._
      setString(Chr(350))
```

```
   StarDesktop.CurrentComponent.CurrentController.getViewCursor()._
      collapseToEnd
End Sub
```

Man könnte es auch mit einem Befehl lösen – allerdings wird dann der Code nicht gerade kürzer:

```
Sub i_ohne_Punkt
   Dim objCursor As Object
   objCursor = _
      StarDesktop.CurrentComponent.CurrentController.getViewCursor()
   objCursor.getText.insertString(objCursor.getStart(), Chr(305), False)
End Sub

Sub I_mit_Punkt
   Dim objCursor As Object
   objCursor = _
      StarDesktop.CurrentComponent.CurrentController.getViewCursor()
   objCursor.getText.insertString(objCursor.getStart(), Chr(304), False)
End Sub

Sub Klein_Yumuschak_g
   Dim objCursor As Object
   objCursor = _
      StarDesktop.CurrentComponent.CurrentController.getViewCursor()
   objCursor.getText.insertString(objCursor.getStart(), Chr(287), False)
End Sub

Sub Gross_Yumuschak_G
   Dim objCursor As Object
   objCursor = _
      StarDesktop.CurrentComponent.CurrentController.getViewCursor()
   objCursor.getText.insertString(objCursor.getStart(), Chr(286), False)
End Sub
```

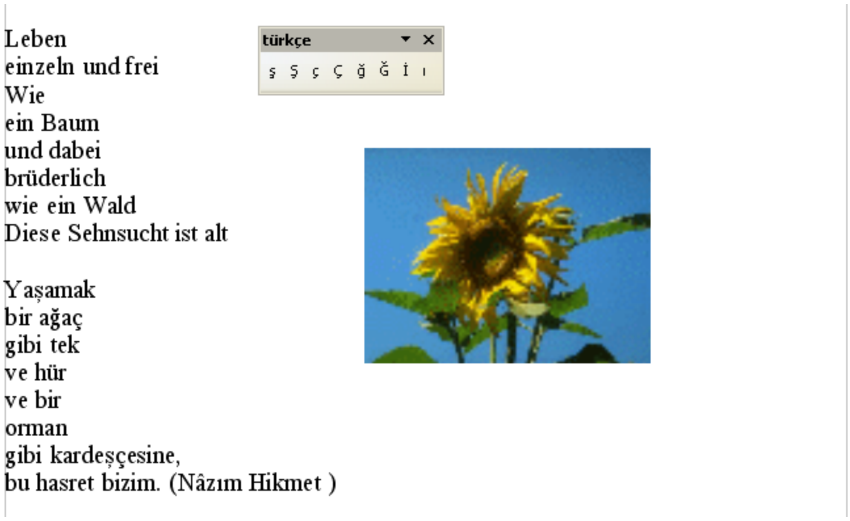


Abbildung 6.114 Nun können nichtdeutschsprachige Texte leicht erstellt werden.

6.7 Markierung auslesen

Manchmal muss der markierte Text ausgelesen werden. Da eine Mehrfachselektion möglich ist, müssen alle möglichen Markierungen durchlaufen werden (*Beispiele06.sxw*):

```
Sub Markierung1
   Dim objDoc As Object
   Dim objMarkierungen As Object
   Dim objMarkierung As Object
   Dim intAnzahl As Integer
   Dim strMarkText As String

   objDoc = ThisComponent
   objMarkierungen = objDoc.getCurrentSelection()
   For intAnzahl = 0 To objMarkierungen.Count - 1
      objMarkierung = objMarkierungen(intAnzahl)
      strMarkText = strMarkText & Chr(13) & objMarkierung.getString()
```

```
    Next

    MsgBox strMarkText

End Sub
```

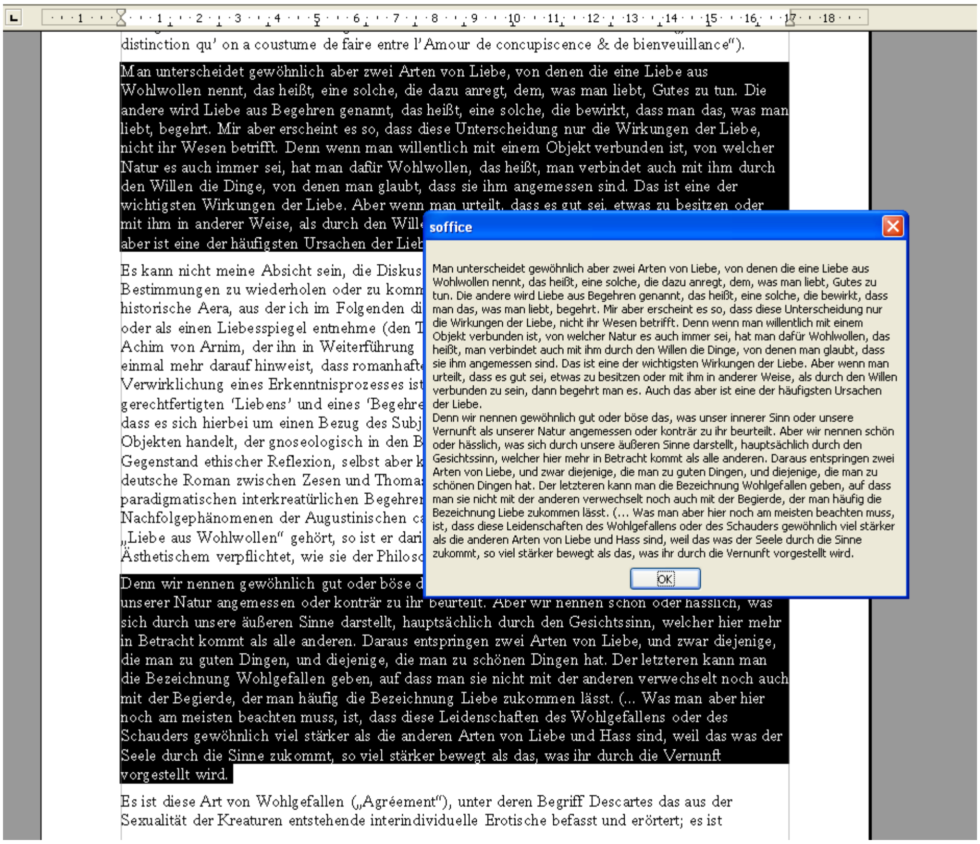


Abbildung 6.115 Der markierte Text

Soll überprüft werden, ob etwas (oder nichts) markiert ist, dann könnte man es wie folgt testen (*Beispiele06.sxw*):

```
Sub Markierung2

    Dim objDoc As Object
```

```
   Dim objMarkierungen As Object
   objDoc = ThisComponent
   objMarkierungen = objDoc.getCurrentSelection()

   If objMarkierungen.Count > 1 Then
      Msgbox "Bitte keine Mehrfachmarkierung!"
   ElseIf Len(objMarkierungen(0).getString()) > 0 Then
      Msgbox "Bitte nichts markieren!"
   End If
End Sub
```

Pitonyak weist zu Recht darauf hin, dass diese Methode unsicher ist, da `getString()` nur einen Text bis maximal 64.000 Zeichen aufnehmen kann – das entspricht etwa 15 Writer-Seiten. Eleganter ist die folgende Methode (*Beispiele06.sxw*):

```
Sub Markierung3
   Dim objDoc As Object
   Dim objMarkierungen As Object
   Dim objMarkierung As Object
   Dim objCursor As Object
   Dim intAnzahl As Integer

   objDoc = ThisComponent
   objMarkierungen = objDoc.getCurrentSelection()
   For intAnzahl = 0 To objMarkierungen.Count - 1
      objMarkierung = objMarkierungen(intAnzahl)
      objCursor = objDoc.Text.createTextCursorByRange(objMarkierung)
      If Not objCursor.IsCollapsed() Then
         MsgBox "Bitte nichts markieren!"
      End If
   Next

End Sub
```

6.8 Texte formatieren

Damit wird deutlich, wie ein Text erzeugt werden kann. Nun wird gezeigt, wie er hart, das heißt direkt beziehungsweise ohne Formatvorlage, formatiert werden kann. Drei zentrale Formatierungskriterien stehen in jeder Textverarbeitung zur Verfügung: Zeichen, Absatz und Seite. Sie werden in den folgenden Kapiteln beschrieben.

6.8.1 Zeichenattribute

Als Zeichenattribute werden all jene Formatattribute bezeichnet, die sich auf einzelne Zeichen beziehen. Dazu gehört:

- Schriftart
- Schriftmodus (fett, kursiv, unterstrichen, durchgestrichen …)
- Schriftgröße
- Schriftfarbe
- Position
- Hintergrundfarbe

Alle diese Elemente müssen den Service „com.sun.star.style.CharacterProperties“ unterstützen. OOo kennt eine ganze Reihe von Services, die diesen Service unterstützen. Dazu zählt der Services „com.sun.star.text.Paragraph“ für Absätze sowie „com.sun.star.text.TextPortion“ für Absatzteile.

Der Service „com.sun.star.style.CharacterProperties“ stellt keinerlei Schnittstellen zur Verfügung, bietet dafür allerdings eine Reihe von Eigenschaften an, über die sich die Zeichenattribute festlegen und auslesen lassen. Eine komplette Liste sämtlicher Zeichenattribute findet sich in der Referenz des OOo API.

Tabelle 6.98 Die wichtigsten Zeichenattribute

Eigenschaft	Wert	Beschreibung
Schrift		
CharFontName	String	Schriftart
CharFontFamily	Konstante	Schriftfamilie gemäß com.sun.star.awt.FontFamily: DONTKNOW (0), DECORATIVE (1), MODERN (2), ROMAN (3), SCRIPT (4), SWISS (5) und SYSTEM (6)
CharFontCharSet	Konstante	Encoding gemäß com.sun.star.awt.CharSetconstant: DONTKNOW, ANSI, MAC, IBMPC_437, IBMPC_850, IBMPC_860, IBMPC_86, IBMPC_863, IBMPC_865, SYS-

Eigenschaft	Wert	Beschreibung
		TEM und SYMBOL
CharWeight	Konstante	Schriftstärke gemäß com.sun.star.awt.FontWeight: DONTKNOW (0), THIN (50), ULTRALIGHT (60), LIGHT (75), SEMILIGHT (90), NORMAL (100), SEMIBOLD (110), BOLD (150), ULTRABOLD (175) und BLACK (200)
CharHeight	Single	Schriftgröße in Punkt (pt)
CharLocale	Struktur	Sprache gemäß com.sun.star.lang.Locale
CharFontType	Konstante	Schrift gemäß com.sun.star.awt.FontType: DONTKNOW (0), RASTER (1), DEVICE (2) und SCALABLE (3)
Schrifteffekt		
CharUnderline	Konstante	Art des Unterstrichs gemäß com.sun.star.awt.FontUnderline: NONE (0), SINGLE (1), DOUBLE (2), DOTTED (3), DONTKNOW (4), DASH (5), LONGDASH (6), DASHDOT (7), DASHDOTDOT (8), SMALLWAVE (9), WAVE (10), DOUBLEWAVE (11), BOLD (12), BOLDDOTTED (13), BOLDDASH (14), BOLDLONGDASH (15), BOLDDASHDOT (16), BOLDDASHDOTDOT (17) und BOLDWAVE (18)
CharUnderlineColor	Long	Farbe des Unterstrichs
CharUnderlineHasColor	Boolean	schaltet die Farbe des Unterstrichs ein oder aus
CharPosture	Konstante	Art des Schriftschnitts gemäß com.sun.star.awt.FontSlant: NONE (0), OBLIQUE (1), ITALIC (2), DONTKNOW (3=, REVERSE_OBLIQUE (4) und REVERSE_ITALIC (5)
CharCaseMap	Konstante	Auszeichnungen gemäß com.sun.star.style.CaseMap: NONE (0), UPPERCASE (1), LOWERCASE (2), TITLE (3) und SMALLCAPS (4)
CharCrossedOut	Boolean	durchgestrichen
CharFlash	Boolean	blinkend
CharStrikeout	Konstante	Durchstreichung gemäß com.sun.star.awt.FontStrikeout: NONE (0), SINGLE (1), DOUBLE (2), DONTKNOW (3), BOLD (4), SLASH (5) und X (6)
CharWordMode	Boolean	wortweise durchgestrichen
CharColor	Long	Farbwert des Textes
CharRelief	Konstante	Relief gemäß com.sun.star.text.FontRelief: NONE (0), EMBOSSED (1) und ENGRAVED (2)
CharContoured	Boolean	Kontur
CharShadowed	Boolean	Schatten
Position		
CharFontPitch	Konstante	Höhe gemäß com.sun.star.awt.FontPitch: DONTKNOW, FI-

Eigenschaft	Wert	Beschreibung
		XED und VARIABLE
CharEscapement	Integer	der prozentuale Wert des Höher- oder Tieferstellens
CharEscapementHeight	Integer	der absolute Wert für hoch- und tiefstellen
CharScaleWidth	Integer	Prozentanteil der Skalierung beim Hoch- oder Tiefstellen
CharRotationIsFitToLine	Boolean	Rotation wird an Zeile angepasst
CharRotation	Integer	Grad der Drehung
CharAutoKerning	Boolean	paarweises Kerning
CharKerning	Integer	der Kerning-Wert (unterschneiden)
Hypertext		
HyperLinkURL	String	URL des Hyperlinks
HyperLinkTarget	String	Hyperlink-Text
HyperLinkName		Hyperlink-Name
Hintergrund		
CharBackColor	Long	Farbwert des Schrifthintergrundes
CharBackTransparent	Boolean	transparenter Hintergrund
Sonstiges		
CharKeepTogether	Boolean	Unterdrückung des automatischen Zeilenumbruches
CharNoHyphenation	Boolean	Unterdrückung der automatischen Silbentrennung
CharNoLineBreak	Boolean	Unterdrückung des Zeilenumbruchs
Vorlage		
CharFontStyleName	String	Name der Zeichenvorlage
CharStyleName	String	Name der Vorlage
VisitedCharStyleName	String	Name der Vorlage des besuchten Hyperlinks
UnvisitedCharStyleName		Name der Vorlage des unbesuchten Hyperlinks

Für asiatische Texte sind interessant: CharCombineIsOn, CharCombinePrefix, CharCombineSuffix, CharEmphasize, RubyText, RubyAdjust, RubyCharStyleName und RubyIsAbove.

Das folgende Beispiel (*Beispiele02.sxw*) erstellt einen Text und verwendet danach unterschiedliche Zeichenformatierungen:

```
Sub Format01
   Dim objDoc As Object
   Dim objCursor As Object
   Dim strName As String
   Dim datBeginn As Date
```

```
Dim docArgs() As New com.sun.star.beans.PropertyValue
Dim strUrl As String

Const ABSATZ As Integer = 0
Const T1 As String = "Zwischen der Schwung GmbH und "
Const T2 As String = " (im Folgenden „Mitarbeiter‟ genannt) " & _
   "wird vereinbart:"
Const T3 As String = "§ 1  Tätigkeit"
Const T4 As String = "Der Mitarbeiter wird mit Wirkung vom "
Const T5 As String = " für das Unternehmen angestellt."
Const T6 As String = "Die Schwung GmbH ist ein international " & _
   "tätiger Konzern mit Standorten im In- und Ausland. Im " & _
   "Hinblick auf die standortübergreifenden Konzern- und " & _
   "Unternehmensaktivitäten erklärt sich der Mitarbeiter bereit, " & _
   "auch in einer anderen Abteilung, einem anderen Betrieb, " & _
   "einem anderen Unternehmensteil oder einem anderen " & _
   "Konzernunternehmen der Schwung GmbH beschäftigt" & _
   " zu werden. Das Unternehmen ist berechtigt, die Aufgaben " & _
   "des Mitarbeiters innerhalb der in § 1 Absatz 1 genannten " & _
   "Tätigkeit zu verändern und dem Mitarbeiter andere Aufgaben," & _
   " ggf. auch an anderen Orten, zuzuweisen, soweit dies dem " & _
   "Mitarbeiter bei Abwägung der betrieblichen und seiner " & _
   "persönlichen Belange zumutbar ist und seinen Kenntnissen und " & _
   "Fähigkeiten entspricht. Das Recht des Unternehmens, dem " & _
   "Mitarbeiter andere Aufgaben zu übertragen, wird auch durch " & _
   "eine langfristige Tätigkeit in einer bestimmten Funktion" & _
   " bzw. an einem bestimmten Arbeitsort nicht eingeschränkt."
Const T7 As String = "Der Mitarbeiter verpflichtet sich:"
Const T8 As String = "a)   allen Anordnungen nach bestem Wissen " & _
   "nachzukommen, alle Arbeiten gewissenhaft und nach bestem " & _
   "Können auszuführen und die Interessen des Unternehmens und " & _
   "der mit ihm verbundenen Unternehmen zu wahren;"
```

```
Const T9 As String = "b)   die Unfallverhütungsvorschriften und " & _
   "sonstige betriebliche Regelungen und Vorschriften zu beachten;"
Const T10 As String = "c) seine ganze Arbeitskraft in den Dienst " & _
  "des Unternehmens zu stellen und die gesamten Ergebnisse " & _
  "seiner Arbeiten sowie seine bei Erfüllung seiner dienstlichen " & _
"Obliegenheiten gemachten oder in den Rahmen der " & _
"wirtschaftlichen, technischen und wissenschaftlichen Betätigung " & _
"des Unternehmens fallenden Beobachtungen, Erkenntnisse und " & _
"Erfahrungen dem zuständigen Vorgesetzten in allen Einzelheiten " & _
"bekannt zu geben und zur Verfügung zu stellen. Das Unternehmen " & _
"ist befugt, diese Arbeiten mit ihren Ergebnissen in jeder ihm " & _
"gut scheinenden Weise zu benutzen und zu verwerten."
Dim LS As New com.sun.star.style.LineSpacing
LS.Mode = com.sun.star.style.LineSpacingMode.MINIMUM
LS.Height = 750

strAnrede = InputBox("Bitte den Namen des Mitarbeiters eingeben:")
datBeginn = InputBox("Bitte den Beginn des Arbeitsverhältnisses " & _
   "eingeben:")
strUrl = "private:factory/swriter"
objDoc = StarDesktop.loadComponentFromURL _
   (strUrl, "_blank", 0, docArgs())
objCursor = objDoc.Text.createTextCursor
objCursor.CharFontName = "Arial"
objCursor.CharHeight = 14
objCursor.CharUnderline = com.sun.star.awt.FontUnderline.SINGLE
objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
objCursor.CharColor = RGB(0, 0, 255)
objCursor.ParaLineSpacing = LS
objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.CENTER

objCursor.String = "Anstellungsvertrag"
```

```
objCursor.gotoEnd(False)
objCursor.CharFontName = "Times New Roman"
objCursor.CharHeight = 12
objCursor.CharUnderline = com.sun.star.awt.FontUnderline.NONE
objCursor.CharWeight = com.sun.star.awt.FontWeight.NORMAL
objCursor.CharColor = RGB(0, 0, 0)

objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
objCursor.String = T1 & strAnrede & T2
objCursor.gotoNextWord(False)
objCursor.gotoNextWord(False)
objCursor.gotoNextWord(True)
objCursor.gotoNextWord(True)
objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD

objCursor.gotoEnd(False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.CharHeight = 14
objCursor.String = T3
objCursor.gotoEnd(False)
objCursor.CharHeight = 12
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = T4 & datBeginn & T5
objCursor.gotoEnd(False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = T6
```

```
    objCursor.gotoEnd(False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objCursor.CharHeight = 14

    objCursor.String = "§ 2    Verpflichtungen des Mitarbeiters"
    objCursor.gotoEnd(False)
    objCursor.CharHeight = 12
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objCursor.ParaLeftMargin = 1000
    objCursor.String = T7
    objCursor.gotoEnd(False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objCursor.ParaLeftMargin = 2000
    objCursor.ParaFirstLineIndent = -1000
    objCursor.String = T8
    objCursor.gotoEnd(False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objCursor.String = T9
    objCursor.gotoEnd(False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objCursor.String = T10
    objCursor.gotoEnd(False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
    objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)

End Sub
```

Anstellungsvertrag

Zwischen der **Schwung GmbH** und Doris Timm (im Folgenden „Mitarbeiter“ genannt) wird vereinbart:

§ 1 Tätigkeit

Der Mitarbeiter wird mit Wirkung vom 01.10.2005 für das Unternehmen angestellt.

Die Schwung GmbH ist ein international tätiger Konzern mit Standorten im In- und Ausland. Im Hinblick auf die standortübergreifenden Konzern- und Unternehmensaktivitäten erklärt sich der Mitarbeiter bereit, auch in einer anderen Abteilung, einem anderen Betrieb, einem anderen Unternehmensteil oder einem anderen Konzernunternehmen der Schwung GmbH beschäftigt zu werden. Das Unternehmen ist berechtigt, die Aufgaben des Mitarbeiters innerhalb der in § 1 Absatz 1 genannten Tätigkeit zu verändern und dem Mitarbeiter andere Aufgaben, ggf. auch an anderen Orten, zuzuweisen, soweit dies dem Mitarbeiter bei Abwägung der betrieblichen und seiner persönlichen Belange zumutbar ist und seinen Kenntnissen und Fähigkeiten entspricht. Das Recht des Unternehmens, dem Mitarbeiter andere Aufgaben zu übertragen, wird auch durch eine langfristige Tätigkeit in einer bestimmten Funktion bzw. an einem bestimmten Arbeitsort nicht eingeschränkt.

§ 2 Verpflichtungen des Mitarbeiters

Der Mitarbeiter verpflichtet sich:

a) allen Anordnungen nach bestem Wissen nachzukommen, alle Arbeiten gewissenhaft und nach bestem Können auszuführen und die Interessen des Unternehmens und der mit ihm verbundenen Unternehmen zu wahren;

b) die Unfallverhütungsvorschriften und sonstige betriebliche Regelungen und Vorschriften zu beachten;

c) seine ganze Arbeitskraft in den Dienst des Unternehmens zu stellen und die gesamten Ergebnisse seiner Arbeiten sowie seine bei Erfüllung seiner dienstlichen Obliegenheiten

Abbildung 6.116 Der fertige, formatierte Text

Das nächste Beispiel (*Beispiele04.sxw*) formatiert alle Ziffern eines markierten Textes tiefer gestellt. Dabei wird die Position um 30% tiefer gestellt, die Schriftgröße wird auf 60% verkleinert:

```
Sub SchriftTieferstellen
   Dim objDatei As Object
   Dim objMarkierung As Object
   Dim objAnfang As Object
   Dim i As Integer

   objDatei = ThisComponent
   objMarkierung = objDatei.getCurrentSelection().getByIndex(0)

   If objDatei.Text.compareRegionStarts(objMarkierung.getEnd(), _
      objMarkierung) >= 0 Then
      objAnfang = objMarkierung.getEnd()
   Else
      objAnfang = objMarkierung.getStart()
   End If

   objCursor = objDatei.Text.createTextCursorByRange(objAnfang)

   For i = 1 To Len(objMarkierung.getString())
      If IsNumeric(Mid(objMarkierung.getString(), i, 1)) Then
         objCursor.goRight (1, True)
         objCursor.CharEscapement = -30
         objCursor.CharEscapementHeight = 60
      Else
         objCursor.goRight (1, False)
      End If
   Next
End Sub
```

Abbildung 6.117 Im markierten Text werden alle Zeichen tiefer gestellt.

Ein paar Bemerkungen zum Code: Da der Benutzer mit gedrückter <Strg>-Taste mehrere Texte markieren kann, muss von der Sammlung sämtlicher markierter Texte (`getCurrentSelection()`) ausgegangen werden. Eigentlich müsste man mit einer Schleife über sämtliche Objekte laufen. Da nicht sicher ist, ob der Benutzer von links nach rechts oder von rechts nach links markiert hat, muss der Beginn herausgefunden werden (`compareRegionStarts`). Nun kann Zeichen für Zeichen des markierten Textes durchlaufen und überprüft werden, ob es sich um eine Ziffer handelt. Der Cursor wird jeweils um eine Position nach rechts verschoben – handelt es sich um eine Zahl, dann wird sie tief gestellt.

6.8.2 Absatzattribute

Als Absatzattribute gelten jene Formatierungsinformationen, die sich nicht auf einzelne Zeichen, sondern auf den gesamten Text zwischen zwei Absatzzeichen beziehen. Die wichtigsten Absatzattribute sind:

- Zeilenabstand
- Ausrichtung
- Einzug und Abstand von oben und unten
- Hintergrundfarbe
- Tabstopps

Über den Service „com.sun.star.style.ParagraphProperties“ sind die Absatzattribute verfügbar. Auch sie stehen in verschiedenen Objekten zur Verfügung. Sämtliche Objekte, die den Service „com.sun.star.text.Paragraph“ unterstützen, bieten auch eine Unterstützung für die Absatzattribute in „com.sun.star.style.ParagraphProperties“ an.

Tabelle 6.99 Die Absatzattribute:

Eigenschaft	Wert	Beschreibung
Einzüge und Abstände		

Eigenschaft	Wert	Beschreibung
ParaLineSpacing	struct	Zeilenabstand (Struktur gemäß com.sun.star.style.-LineSpacing). Die Height wird als Short-Wert angegeben. Die Eigenschaft LineSpacingMode wird als Konstante angegeben: PROP (0): Wert der Höhe als proportionaler Wert MINIMUM (1): minimaler Zeilenabstand LEADING (2): Abstand zur vorhergehenden Zeile FIX (3) exakter Abstand
ParaLeftMargin	Long	linker Einzug in 100stel Millimetern
ParaRightMargin	Long	rechter Einzug in 100stel Millimetern
ParaTopMargin	Long	oberer Abstand in 100stel Millimetern
ParaBottomMargin	Long	unterer Abstand in 100stel Millimetern
ParaFirstLineIndent	Long	Erstzeileneinzug
ParaIsAutoFirstLineIndent	Boolean	automatischer Erstzeileneinzug
ParaRegisterModeActive	Boolean	Registerhaltigkeit
Ausrichtung		
ParaAdjust	enum	vertikale Textausrichtung (Konstanten gemäß com.sun.star.style.ParagraphAdjust: LEFT (0), RIGHT (1), BLOCK (2), CENTER (3) und STRECH (4)
ParaLastLineAdjust	Integer	richtet die letzte Zeile eines Absatzes als Block aus
ParaExpandSingleWord	Boolean	einzelnes Wort „austreiben“
ParaVertAlignment	enum	Text-an-Text-Ausrichtung (Konstanten gemäß com.sun.star.text.ParagraphVertAlign AUTOMATIC (0), BASELINE (1), TOP (2), CENTER (3) und BOTTOM (4)
Textfluss		
ParaIsHyphenation	Boolean	automatische Silbentrennung
ParaHyphenationMaxHyphens	Integer	max. aufeinander folgende Trennstellen
ParaHyphenationMaxLeadingChars	Integer	max. Zeichen am Zeilenende
ParaHyphenationMaxTrailingChars	Integer	max. Zeichen am Zeilenanfang
PageDescName	String	Seitenumbruch mit Namen der Vorlage der neuen Seite
PageNumberOffset	Integer	Nummer der neuen Seite
ParaKeepTogether	Boolean	Absatz nicht trennen

Eigenschaft	Wert	Beschreibung
ParaSplit	Boolean	Absätze zusammenhalten
ParaOrphans	Integer	Schusterjungenregelung
ParaWidows	Integer	Hurenkinderregelung
Nummerierung		
ParaLineNumberCount	Boolean	Zeilennummerierung
NumberingIsNumber	Boolean	Absatznummerierung verwendet Zahl (und kein Symbol)
ParaLineNumberStartValue	Long	erste Nummer der Zeilennummerierung
NumberingLevel	Integer	Nummer der Ebene der Absatznummerierung
ParaIsNumberingRestart	Boolean	Nummerierung beginnt an diesem Absatz
NumberingStartValue	Integer	Nummer des Beginns der Absatznummerierung
NumberingRules	Enum	Nummerierung (com.sun.star.container.XIndexRe-place)
Tabulator		
ParaTabStops	Array of struct	Art und Position der Tabulatoren (Array mit Struktu-ren des Typs com.sun.star.style.TabStop)
Hintergrund		
ParaBackColor	Long	Hintergrundfarbe
ParaBackTransparent	Boolean	der Hintergrund wird auf transparent gesetzt
ParaBackGraphicURL	String	URL des Hintergrundbildes
ParaBackGraphicFilter	String	zugehöriger Filter
ParaBackGraphicLocation	Enum	die Ausrichtung (Konstanten gemäß com.sun.star.style.GraphicLocation: NONE, LEFT_TOP, MIDDLE_TOP, RIGHT_TOP, LEFT_MIDDLE, MIDDLE_MIDDLE, RIGHT_MIDD-LE, LEFT_BOTTOM, MIDDLE_BOTTOM, RIGHT_BOTTOM, AREA, TI-LED)
Umrandung		
ParaShadowFormat	Array of struct	bestimmt das Linienformat nach com.sun.star.ta-ble.ShadowFormat: Location (Konstanten gemäß com.sun.star.ta-ble.ShadowLocation: NONE, TOP_LEFT, TOP_RIGHT, BOTTOM_LEFT, BOTTOM_RIGHT) ShadowWidth (integer) IsTransparent (Boolean)

Eigenschaft	Wert	Beschreibung
		Color (Long)
LeftBorder, RightBorder, TopBorder, BottomBorder	Array of struct	bestimmt die jeweilige Linie nach com.sun.star.table.BorderLine: Color (Integer) InnerLineWidth (Double) OuterLineWidth (Double) LineDistance (Double)
BorderDistance	Long	Abstand zwischen Absatz zum „Inhalt“
LeftBorderDistance, RightBorderDistance, TopBorderDistance, BottomBorderDistance	Long	Abstand der Linie zum „Inhalt“
ParaIsConnectBorder	Boolean	Synchronisieren
Initialen		
DropCapFormat	Array of struct	Initialen anzeigen (Array mit Strukturen des Typs com.sun.star..style.DropCapFormat): Lines, Count, Distance
DropCapWholeWord	Boolean	ganzes Wort als Initiale zeigen
Einfügen/Manueller Umbruch		
BreakType	Enum	Umbruch vor dem Absatz des Typs com.sun.star.style.BreakType: NONE, COLUMN_BEFORE, COLUMN_AFTER, COLUMN_BOTH, PAGE_BEFORE, PAGE_AFTER, PAGE_BOTH
Vorlagen		
ParaStyleName	String	Name der Absatzvorlage
NumberingStyleName	String	Name der Nummerierungsvorlage
DropCapCharStyleName	String	Name der Zeichenvorlage der Initialen
ParaUserDefinedAttributes	Array of struct	speichert XML-Attribute (Array mit Strukturen des Typs com.sun.star.container.XNameContainer)

Das folgende Programm (*Beispiele02.sxw*) fügt nicht nur Text ein und formatiert die Zeichen, sondern auch die Absätze (siehe Abbildung 6.10):

```
[...]
   Dim LS As New com.sun.star.style.LineSpacing
   LS.Mode = com.sun.star.style.LineSpacingMode.MINIMUM
```

```
   LS.Height = 750

   strAnrede = InputBox("Bitte den Namen des Mitarbeiters eingeben:")
   datBeginn = InputBox("Bitte den Beginn des Arbeitsverhältnisses " & _
      eingeben:")

   objDoc = ThisComponent
   objCursor = objDoc.Text.createTextCursor
[...]
   objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.CENTER

   objCursor.String = "Anstellungsvertrag"

   objCursor.gotoEnd(False)
[...]
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
   objCursor.String = T1 & strAnrede & T2
[...]

   objCursor.String = "§ 2    Verpflichtungen des Mitarbeiters"
   objCursor.gotoEnd(False)
   objCursor.CharHeight = 12
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.ParaLeftMargin = 1000
   objCursor.String = T7
   objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.ParaLeftMargin = 2000
```

```
   objCursor.ParaFirstLineIndent = -1000
   objCursor.String = T8
[...]

End Sub
```

Das nächste Beispiel (*Beispiele05.sxw*) formatiert einen bereits erstellten Text, indem es überprüft, ob die einzelnen Elemente Absätze sind. Falls die Zeichenlänge nicht 30 übersteigt, wird die Zeile als Überschrift in der Future, 14 pt, zentriert mit einem Abstand nach 1 cm formatiert. Der Fließtext dagegen wird in der Garamond, 12 pt, Blicksatz mit einem nachfolgenden Abstand von 0,5 cm formatiert. Für ihn wird die Silbentrennung eingeschaltet. Beide Absatztypen werden zweizeilig formatiert. Beachten Sie, dass Sie die Eigenschaft `LineSpacing.Height` – anders als die übrigen Absatz- und Zeichenformatierungen – nicht direkt zuweisen können, sondern über ein LineSpacing-Objekt. Enthält die Zeile den Begriff „Jamie Oliver“, wird sie kursiv formatiert, enthält sie einen Tabulator, wird sie mit einem hängenden Einzug versehen:

```
Sub TextInForm

   Dim objDoc As Object
   Dim objEnum As Object
   Dim objTextElement As Object
   Dim objZeilenabstand As New com.sun.star.style.LineSpacing
   objDoc = ThisComponent
   objEnum = objDoc.Text.createEnumeration

   objZeilenabstand.Mode = com.sun.star.style.LineSpacingMode.FIX
   objZeilenabstand.Height = 750

   Do While objEnum.hasMoreElements
      objTextElement = objEnum.nextElement
      If objTextElement. _
         supportsService("com.sun.star.text.Paragraph") Then
         If objTextElement.String <> "" Then
            If Len(objTextElement.String) < 30 Then
               With objTextElement
                  .ParaAdjust = com.sun.star.style.ParagraphAdjust.CENTER
```

```
                .CharFontName = "Futura Md Bt"
                .CharHeight = 14
                .ParaLineSpacing = objZeilenabstand
                .ParaBottomMargin = 1000
             End With
          Else
             With objTextElement
                .ParaAdjust = com.sun.star.style.ParagraphAdjust.BLOCK
                .CharFontName = "Garamond"
                .CharHeight = 12
                .ParaLineSpacing = objZeilenabstand
                .ParaBottomMargin = 500
                .ParaIsHyphenation = True
             End With
             If InStr(objTextElement.String, Chr(9)) Then
                With objTextElement
                   .ParaFirstLineIndent = -2000
                   .ParaLeftMargin = 2000
                End With
             End If
             If InStr(objTextElement.String, "Jamie Oliver") Then
                objTextElement.CharPosture = _
                   com.sun.star.awt.FontSlant.ITALIC
             End If
          End If
       End If
    End If
 Loop

End Sub
```

Schoko-Mousse
4 Portionen
Zutaten: 225 gute Zartbitterschokolade, zerstoßen, 70 g Butter, in Stücke geschnitten, 350 g Sahne oder Crème double, 2 große Eier, möglichst Bio-Eier, 5 cm² Zitronenschale, 1 – 1 ½ EL Amaretto, 2 EL guter Honig
Schokolade und Butter in einer Schüssel über köchelndem Wasser langsam schmelzen lassen und von der Kochstelle nehmen. In einer anderen Schüssel die Sahne halb steif schlagen (sie darf noch nicht zu steif sein). In einer dritten Schüssel Eier und Honig schlagen, bis eine leicht schaumige Masse entstanden ist, dann den Amaretto, die flüssige Schoko-Butter-Masse, die Zitronenschale und die Sahne vorsichtig darunter heben, so dass sie nicht zusammenfällt. In kleine Weingläser oder Dessertschalen füllen und vor dem Servieren mindestens eine Stunde kalt stellen. Mit Sesam-Toffee-Bissen servieren.
Aus: „Genial kochen mit Jamie Oliver", S. 279

Abbildung 6.118 Vorher …

Schoko-Mousse

4 Portionen

Zutaten: 225 gute Zartbitterschokolade, zerstoßen, 70 g Butter, in Stücke geschnitten, 350 g Sahne oder Crème double, 2 große Eier, möglichst Bio-Eier, 5 cm² Zitronenschale, 1 – 1 ½ EL Amaretto, 2 EL guter Honig

Schokolade und Butter in einer Schüssel über köchelndem Wasser langsam schmelzen lassen und von der Kochstelle nehmen. In einer anderen Schüssel die Sahne halb steif schlagen (sie darf noch nicht zu steif sein). In einer dritten Schüssel Eier und Honig schlagen, bis eine leicht schaumige Masse entstanden ist, dann den Amaretto, die flüssige Schoko-Butter-Masse, die Zitronenschale und die Sahne vorsichtig darunter heben, so dass sie nicht zusammenfällt. In kleine Weingläser oder Dessertschalen füllen und vor dem Servieren mindestens eine Stunde kalt stellen. Mit Sesam-Toffee-Bissen servieren.

Aus: „Genial kochen mit Jamie Oliver", S. 279

Abbildung 6.119 … und nachher

6.8.3 Tabulatoren

Tabelle 6.100 Die Struktur TabStop verwendet folgende Elemente:

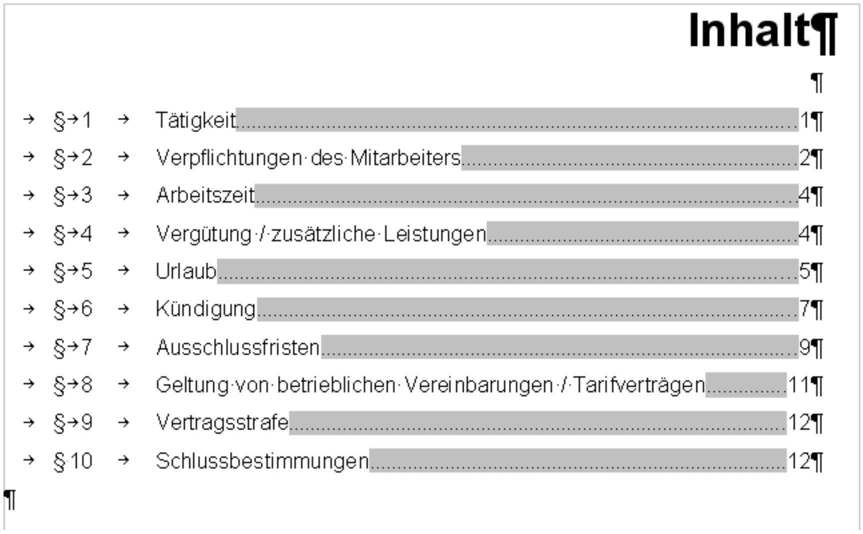
Element	Wert	Beschreibung
Position	Long	die Tabulatorposition in 100stel mm
Alignment	Enum	Ausrichtung des Tabulators gemäß com.sun.star.style.TabAlign (LEFT, CENTER, RIGHT, DECIMAL, DEFAULT)
DecimalChar	Char	bestimmt das Zeichen für den dezimalen Tabulator
FillChar	Char	das Füllzeichen (anzugeben als ASCII-Code)

Zwar sind Tabulatoren auch Eigenschaften von Absatzformatierungen, da sie jedoch als einziges Attribut über eine beliebige Anzahl von Elementen verfügen können, muss ein Datenfeld deklariert und gefüllt werden. Dann kann ihm die Eigenschaften „Position“, „Alignment“, „DecimalChar“ und „FillChar“ zugewiesen werden. Dies soll an einem kleinen Beispiel (*Beispiele02.sxw*) demonstriert werden:

```
Sub Inhaltsverzeichnis
   Dim objDoc As Object
   Dim objCursor As Object
   Dim docArgs() As New com.sun.star.beans.PropertyValue
   Dim strUrl As String

   Const ABSATZ As Integer = 0
   Const T1 As String = "    §      1      Tätigkeit      1"
   Const T2 As String = "    §      2      Verpflichtungen des " & _
      "Mitarbeiters   2"
   Const T3 As String = "    §      3      Arbeitszeit    4"
   Const T4 As String = "    §      4      Vergütung/zusätzliche " & _
      "Leistungen     4"
   Const T5 As String = "    §      5      Urlaub  5"
   Const T6 As String = "    §      6      Kündigung      7"
   Const T7 As String = "    §      7      Ausschlussfristen     9"
   Const T8 As String = "    §      8      Geltung von betrieblichen " & _
      "Vereinbarungen/Tarifverträgen 11"
```

```
Const T9 As String = "    §      9      Vertragsstrafe 12"
Const T10 As String = "   §      10     Schlussbestimmungen   12"

Dim LS As New com.sun.star.style.LineSpacing
Dim Tab(3) As New com.sun.star.style.TabStop

LS.Mode = com.sun.star.style.LineSpacingMode.MINIMUM
LS.Height = 750
Tab(0).Position = 1000
Tab(0).Alignment = com.sun.star.style.TabAlign.LEFT
Tab(1).Position = 1750
Tab(1).Alignment = com.sun.star.style.TabAlign.RIGHT
Tab(2).Position = 3000
Tab(2).Alignment = com.sun.star.style.TabAlign.LEFT
Tab(3).Position = 16000
Tab(3).Alignment = com.sun.star.style.TabAlign.RIGHT
Tab(3).FillChar = 46

strUrl = "private:factory/swriter"
objDoc = StarDesktop.loadComponentFromURL _
   (strUrl, "_blank", 0, docArgs())

objCursor = objDoc.Text.createTextCursor
objCursor.CharFontName = "Arial"
objCursor.CharHeight = 26
objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
objCursor.ParaLineSpacing = LS
objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.RIGHT
objCursor.ParaRightMargin = 1000
objCursor.String = "Inhalt"

objCursor.gotoEnd(False)
```

```
   objCursor.CharHeight = 12
   objCursor.CharWeight = com.sun.star.awt.FontWeight.NORMAL

   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
   objCursor.ParaTabStops = Tab()
   objCursor.String = T1
   objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T2:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T3:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T4:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T5:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T6:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T7:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T8:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T9:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = T10:    objCursor.gotoEnd(False)
   objDoc.Text.insertControlCharacter(objCursor, ABSATZ, False)
End Sub
```


Inhalt¶

¶

→ §→1 → Tätigkeit .... 1¶
→ §→2 → Verpflichtungen·des·Mitarbeiters .... 2¶
→ §→3 → Arbeitszeit .... 4¶
→ §→4 → Vergütung·/·zusätzliche·Leistungen .... 4¶
→ §→5 → Urlaub .... 5¶
→ §→6 → Kündigung .... 7¶
→ §→7 → Ausschlussfristen .... 9¶
→ §→8 → Geltung·von·betrieblichen·Vereinbarungen·/·Tarifverträgen .... 11¶
→ §→9 → Vertragsstrafe .... 12¶
→ §10 → Schlussbestimmungen .... 12¶

¶

Abbildung 6.120 Text mit Tabulatoren wird eingefügt und formatiert.

6.9 Seitenattribute

Unter Seitenattributen versteht man sämtliche Formatierungsangaben, die sich auf die Positionierung des Dokumentinhaltes auf einer Seite beziehen sowie auf optische Elemente, die sich Seite für Seite wiederholen. Hierzu zählen unter anderem:

- das zugrunde liegende Papierformat,
- der Seitenrand und
- die Kopf- und Fußzeilen.

Das Vorgehen bei der Definition von Seitenattributen unterscheidet sich von den übrigen Formatierungen. Während der Anwender bei Zellen, Absätzen und Zeichenelementen die Wahl hat, ob er seine Formatierungen direkt an einem bestimmten Dokumentteil anwendet oder indirekt über den Einsatz von Vorlagen festlegt, können Seitenattribute prinzipiell nur über die zugehörige Seitenvorlage definiert werden. Beim Einfügen einer Kopf- oder Fußzeile muss daher beispielsweise nicht das eigentliche Dokument, sondern die verwendete Seitenvorlage um eine Kopf- beziehungsweise Fußzeile erweitert werden.

Nachfolgend findet sich eine Beschreibung der zentralen Attribute zur Formatierung von Dokumentseiten.

Ein großer Teil der beschriebenen Vorlagen steht nicht nur innerhalb von Tabellendokumenten zur Verfügung, sondern auch innerhalb von Textdokumenten. Die für beide Dokumentarten gültigen Seitenattribute sind in dem Service „com.sun.star.style.PageProperties“ definiert.

In OOo erfolgt die Definition der Seitenattribute über eine Seitenvorlage, die wiederum mit dem zugehörigen Dokument verknüpft wird.

6.9.1 Seitenhintergrund

Tabelle 6.101 Die Eigenschaften von „com.sun.star.style.PageProperties“ für den Seitenhintergrund

Eigenschaft	Wert	Beschreibung
BackColor	long	Farbe des Hintergrunds
BackGraphicURL	String	URL der zu verwendenden Hintergrundgrafik
BackGraphicFilter	String	Name des Filters zur Interpretation der Hintergrundgrafik
BackGraphicLocation	Enum	Position der Hintergrundgrafik (Wert laut Aufzählung com.sun.star.style.GraphicLocation)
BackTransparent	Boolean	stellt den Hintergrund transparent dar

6.9.2 Seitenformat

Tabelle 6.102 Die Eigenschaften von „com.sun.star.style.PageProperties“ für das Seitenformat:

Eigenschaft	Wert	Beschreibung
IsLandscape	Boolean	Ausgabe im Querformat
Width	long	Breite der Seite in 100stel Millimetern
Height	long	Höhe der Seite in 100stel Millimetern
PrinterPaperTray	String	Name des zu verwendenden Drucker-Ausgabeschachtes

Das folgende Beispiel (*Beispiele03.sxw*) setzt die Seitengröße der Seitenvorlage „Default“ auf die Werte für das Format DIN A5 quer (Höhe 14,8 cm, Breite 21 cm):

```
Sub SeiteEinrichten01
```

```
   Dim objDatei As Object
   Dim objStyleFamilies As Object
   Dim objSeitenvorlage As Object
   Dim objDefPage As Object
   Dim docArgs() As New com.sun.star.beans.PropertyValue
   Dim strUrl As String

   strUrl = "private:factory/swriter"
   objDatei = StarDesktop.loadComponentFromURL _
      (strUrl, "_blank", 0, docArgs())

   objStyleFamilies = objDatei.StyleFamilies
   objSeitenvorlage = objStyleFamilies.getByName("PageStyles")
   objDefPage = objSeitenvorlage.getByName("Standard")
   objDefPage.IsLandscape = True
   objDefPage.Width = 21000
   objDefPage.Height = 14800

End Sub
```

Abbildung 6.121 Die Seite wird eingerichtet.

6.9.3 Seitenrand, -rahmen und -schatten

Zur Anpassung des Seitenrandes sowie eventueller Rahmen und Schatten steht der Service „com.sun.star.style.PageProperties" bereit:

Tabelle 6.103 com.sun.star.style.PageProperties verfügt über folgende Eigenschaften

Eigenschaft	**Wert**	**Beschreibung**
LeftMargin	Long	linker Seitenrand in 100stel Millimetern
RightMargin	Long	rechter Seitenrand in 100stel Millimetern
TopMargin	Long	oberer Seitenrand in 100stel Millimetern
BottomMargin	Long	unterer Seitenrand in 100stel Millimetern
LeftBorder	Struct	Angaben zur linken Linie des Seitenrahmens (Struktur com.sun.star.table.BorderLine: Color, InnerLineWidth, OuterLineWidth und LineDistance)

Eigenschaft	Wert	Beschreibung
RightBorder	Struct	Angaben zur rechten Linie des Seitenrahmens (Struktur com.sun.star.table.BorderLine)
TopBorder	Struct	Angaben zur oberen Linie des Seitenrahmens (Struktur com.sun.star.table.BorderLine)
BottomBorder	Struct	Angaben zur unteren Linie des Seitenrahmens (Struktur com.sun.star.table.BorderLine)
LeftBorderDistance	Long	Abstand zwischen linkem Seitenrahmen und Seiteninhalt in 100stel Millimetern
RightBorderDistance	Long	Abstand zwischen rechtem Seitenrahmen und Seitenin-halt in 100stel Millimetern
TopBorderDistance	Long	Abstand zwischen oberem Seitenrahmen und Seiteninhalt in 100stel Millimetern
BottomBorderDistance	Long	Abstand zwischen unterem Seitenrahmen und Seitenin-halt in 100stel Millimetern
ShadowFormat	Struct	Angaben zum Schatten des Content-Bereiches der Seite (Struktur com.sun.star.table.ShadowFormat: Location, ShadowWidth, IsTransparent und Color)

Das folgende Beispiel (*Beispiele03.sxw*) setzt den linken, rechten oberen und unteren Rand der Seitenvorlage „Default“ auf 2 Zentimeter und zeichnet um das Tabellenblatt eine doppelte rote Linie im Abstand von einem Millimeter:

```
Sub SeiteEinrichten02
   Dim objDatei As Object
   Dim objStyleFamilies As Object
   Dim objSeitenvorlage As Object
   Dim objDefPage As Object
   Dim objRandLinie As New com.sun.star.table.BorderLine
   Dim docArgs() As New com.sun.star.beans.PropertyValue
   Dim strUrl As String

   strUrl = "private:factory/swriter"
   objDatei = StarDesktop.loadComponentFromURL _
      (strUrl, "_blank", 0, docArgs())
```

```
    objStyleFamilies = objDatei.StyleFamilies
    objSeitenvorlage = objStyleFamilies.getByName("PageStyles")
    objDefPage = objSeitenvorlage.getByName("Standard")

    objDefPage.LeftMargin = 2000
    objDefPage.RightMargin = 2000
    objDefPage.TopMargin = 2000
    objDefPage.BottomMargin = 2000
    objRandLinie.Color = RGB(255, 0, 0)
    objRandLinie.InnerLineWidth = 1
    objRandLinie.OuterLineWidth = 1
    objRandLinie.LineDistance = 100

    objDefPage.LeftBorder = objRandLinie
    objDefPage.RightBorder = objRandLinie
    objDefPage.TopBorder = objRandLinie
    objDefPage.BottomBorder = objRandLinie

End Sub
```

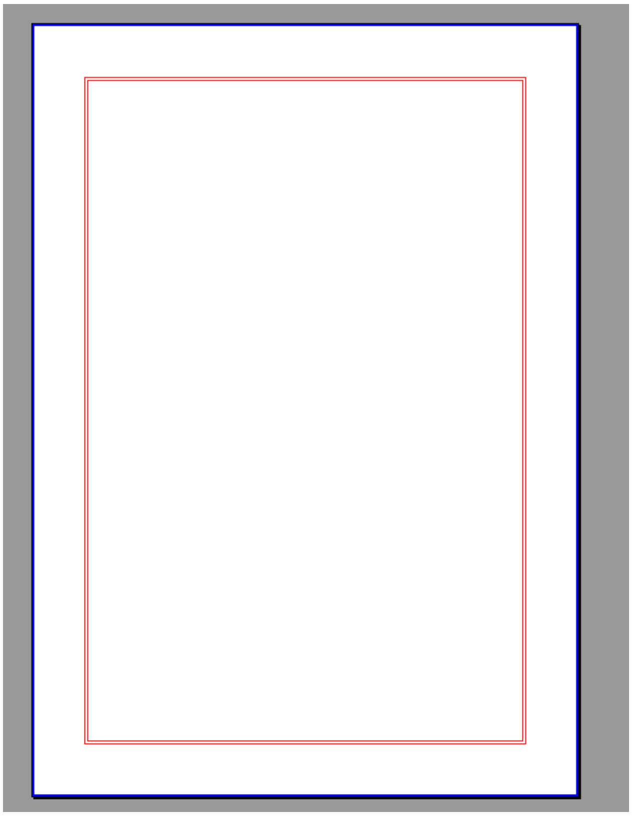


Abbildung 6.122 Ein anderes Seitenformat

6.9.4 Kopf- und Fußzeilen

Die Kopf- und Fußzeilen eines Dokumentes gehören zu den Seitenattributen und werden ebenfalls über den Service „com.sun.star.style.PageProperties“ definiert.

Tabelle 6.104 Die wichtigsten Eigenschaften für Kopfzeilen und Fußzeilen lauten:

Eigenschaft	**Wert**	**Beschreibung**
HeaderIsOn/FooterIsOn	Boolean	Kopfzeile/Fußzeile ist aktiviert
HeaderLeftMargin/FooterLeftMargin	Long	Abstand zwischen Kopfzeile/Fußzeile und linkem Seitenrand in 100stel Millimeter
HeaderRightMargin/FooterRightMargin	Long	Abstand zwischen Kopfzeile/Fußzeile und rech-

Eigenschaft	Wert	Beschreibung
		tem Seitenrand in 100stel Millimetern
HeaderBodyDistance/ FooterBodyDistance	Long	Abstand zwischen Kopfzeile/Fußzeile und Hauptbereich des Dokumentes in 100stel Millimetern
HeaderHeight/FooterHeight	Long	Höhe der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderIsDynamicHeight/ FooterIsDynamicHeight	Boolean	die Höhe der Kopfzeile/Fußzeile wird automatisch an den Inhalt angepasst
HeaderLeftBorder/FooterLeftBorder	Struct	Angaben zur linken Linie des Rahmens um die Kopfzeile/Fußzeile (Struktur com.sun.star.table.BorderLine: Color, InnerLineWidth, OuterLineWidth und LineDistance)
HeaderRightBorder/FooterRightBorder	Struct	Angaben zur rechten Linie des Rahmens um die Kopfzeile/Fußzeile (Struktur com.sun.star.table.BorderLine)
HeaderTopBorder/FooterTopBorder	Struct	Angaben zur oberen Linie des Rahmens um die Kopfzeile/Fußzeile (Struktur com.sun.star.table.BorderLine)
HeaderBottomBorder/FooterBottomBorder	Struct	Angaben zur unteren Linie des Rahmens um die Kopfzeile/Fußzeile (Struktur com.sun.star.table.BorderLine)
HeaderLeftBorderDistance	Long	Abstand zwischen linker Rahmenlinie und Inhalt der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderRightBorderDistance/ FooterRightBorderDistance	Long	Abstand zwischen rechter Rahmenlinie und Inhalt der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderTopBorderDistance/ FooterTopBorderDistance	Long	Abstand zwischen oberer Rahmenlinie und Inhalt der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderBottomBorderDistance/ FooterBottomBorderDistance	Long	Abstand zwischen unterer Rahmenlinie und Inhalt der Kopfzeile/Fußzeile in 100stel Millimetern
HeaderIsShared/FooterIsShared	Boolean	die Kopfzeilen/Fußzeilen auf den linken und rechten Seiten enthalten den gleichen Inhalt (siehe HeaderText, HeaderTextLeft und HeaderTextRight bzw. FooterText, FooterTextLeft und FooterTextRight)
HeaderBackColor/FooterBackColor	Long	Hintergrundfarbe der Kopfzeile/Fußzeile
HeaderBackGraphicURL/	String	URL der zu verwendenden Hintergrundgrafik

Eigenschaft	Wert	Beschreibung
FooterBackGraphicURL		
HeaderBackGraphicFilter/ FooterBackGraphicFilter	String	Name des Filters zur Interpretation der Hintergrundgrafik für die Kopfzeile/Fußzeile
HeaderBackGraphicLocation/ FooterBackGraphicLocation	Enum	Position der Hintergrundgrafik für die Kopfzeile/Fußzeile (Wert laut Aufzählung com.sun.star.style.GraphicLocation)
HeaderBackTransparent/ FooterBackTransparent	Boolean	zeigt den Hintergrund der Kopfzeile/Fußzeile transparent an
HeaderShadowFormat/ FooterShadowFormat	Struct	Angaben zum Schatten der Kopfzeile/Fußzeile (Struktur com.sun.star.table.ShadowFormat)

6.9.5 Texte von Kopf- und Fußzeilen ändern

Für Textdokumente (OOo Writer) existiert ein anderer Mechanismus zum Ändern der Kopf- und Fußzeilentexte als in StarCalc, da diese dort aus einem einzigen Textblock bestehen.

Tabelle 6.105 com.sun.star.style.PageProperties verwendet folgende Eigenschaften:

Eigenschaft	Wert	Beschreibung
HeaderText	Object	Text-Objekt mit Inhalt der Kopfzeile (Service com.sun.star.text.XText)
HeaderTextLeft	Object	Text-Objekt mit Inhalt der Kopfzeilen auf den linken Seiten (Service com.sun.star.text.XText)
HeaderTextRight	Object	Text-Objekt mit Inhalt der Kopfzeilen auf den rechten Seiten (Service com.sun.star.text.XText)
FooterText	Object	Text-Objekt mit Inhalt der Fußzeile (Service com.sun.star.text.XText).
FooterTextLeft	Object	Objekt mit Inhalt der Fußzeilen auf den linken Seiten (Service com.sun.star.text.XText)
FooterTextRight	Object	Text-Objekt mit Inhalt der Fußzeilen auf den rechten Seiten (Service com.sun.star.text.XText)

Das folgende Beispiel (*Beispiele03.sxw*) legt in der Vorlage „Standard“ eines Textdokumentes eine Kopfzeile an, in die der Text „Gedichte“ geschrieben wird.

```
Sub SeiteEinrichten03
```

```
    Dim objDatei As Object
    Dim objStyleFamilies As Object
    Dim objSeitenvorlage As Object
    Dim objDefPage As Object
    Dim objHText As Object
    Dim objHInhalt As Object
    Dim objCursor As Object
    Dim docArgs() As New com.sun.star.beans.PropertyValue
    Dim strUrl As String

    strUrl = "private:factory/swriter"
    objDatei = StarDesktop.loadComponentFromURL _
       (strUrl, "_blank", 0, docArgs())

    objStyleFamilies = objDatei.StyleFamilies
    objSeitenvorlage = objStyleFamilies.getByName("PageStyles")
    objDefPage = objSeitenvorlage.getByName("Standard")

    objDefPage.HeaderIsOn = True
    objHText = objDefPage.HeaderText

    objCursor = objHText.Text.createTextCursor
    objCursor.CharFontName = "Edwardian Script ITC"
    objCursor.CharHeight = 48
    objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
    objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.CENTER
    objHText.String = "Gedichte"

End Sub
```

Der Zugriff erfolgt in diesem Falle anstelle des HeaderFooterContent-Objektes direkt über die Eigenschaft `HeaderText` der Seitenvorlage. Damit wird nun klar, wie man die Kopfzeile formatieren kann:

```
[...]
```

```
   objCursor = objHText.Text.createTextCursor
   objCursor.CharFontName = "Edwardian Script ITC"
   objCursor.CharHeight = 48
   objCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
   objCursor.ParaAdjust = com.sun.star.style.ParagraphAdjust.CENTER
   objHText.String = "Gedichte"
[...]
```

Abbildung 6.123 Das Seitenformat mit formatierter Kopfzeile

6.10 Formatvorlagen

Neben der direkten (oder harten) Formatierung ist eine Formatierung über die Verwendung von Vorlagen (Stilen) möglich. Diese wird als indirekte (oder weiche) Formatierung bezeichnet. Hierbei weist der Anwender der betreffenden Textstelle anstelle der direkten Formatattribute eine Vorlage zu. Soll das Layout des Textes nachträglich geändert werden, so braucht der Anwender lediglich die Vorlage anzupassen.

OOo ändert daraufhin die Darstellung sämtlicher Textstellen, bei denen die Vorlage eingesetzt wird. Das nächste Beispiel (*Beispiele04.sxw*) erzeugt eine neue Absatzvorlage. Beachten Sie, dass sie mit der Methode `insertByName` an die Sammlung „StyleFamilies“ angehängt werden muss:

```
Sub NeueVorlage
   Dim objDatei As Object
   Dim objStyleFamilies As Object
   Dim objAbsatzvorlagen As Object
   Dim objAbsatzvorlage As Object
```

```
    Dim docArgs() As New com.sun.star.beans.PropertyValue
    Dim strUrl As String

    strUrl = "private:factory/swriter"
    objDatei = StarDesktop.loadComponentFromURL _
        (strUrl, "_blank", 0, docArgs())

    objStyleFamilies = objDatei.StyleFamilies
    objAbsatzvorlagen = objStyleFamilies.getByName("ParagraphStyles")
    If objAbsatzvorlagen.hasByName("Fließtext") = False Then
        objAbsatzvorlage = _
        ThisComponent.createInstance("com.sun.star.style.ParagraphStyle")
        objAbsatzvorlage.setname("Fließtext")
        objAbsatzvorlagen.insertByName("Fließtext", objAbsatzvorlage)
    End If
End Sub
```

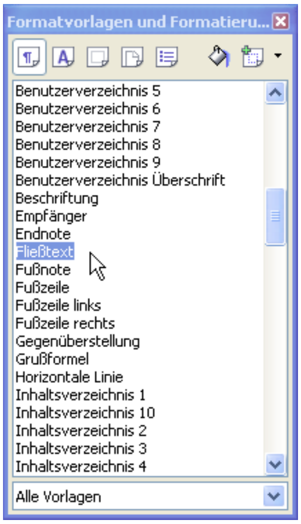


Abbildung 6.124 Die neue Vorlage ist in der Liste der Formatvorlagen (in OOo 1.4 im „Stylist") sichtbar.

Die geöffnete Liste der Formatvorlagen (in OOo 1.1 „Stylist“) zeigt Ihnen nicht in der Kategorie „automatisch“ die aktuelle Liste der Vorlagen an – die Liste muss geschlossen und erneut geöffnet werden, oder Sie müssen in die Kategorie „Benutzerdefiniert“ beziehungsweise „alle Vorlagen“ wechseln.

Wenn Sie eine Vorlage erzeugen wie eine, die bereits existiert, dann wird eine Fehlermeldung erzeugt. Jedoch kann man mit der Methode `hasByName` überprüfen, ob es bereits eine solche Vorlage gibt:

```
   If objAbsatzvorlagen.hasByName("Fließtext") = False Then
```

Mit diesem Wissen ist es nun nicht mehr sehr schwierig, einen Dialog zu erstellen, in den der Benutzer Daten eingibt (*Beispiele04.sxw*):

```
Option Explicit


Dim Dlg As Object


Sub DialogStarten
   DialogLibraries.LoadLibrary("Standard")
   Dlg = CreateUnoDialog(DialogLibraries.Standard.dlgBrief)
   Dlg.Execute()
End Sub
```

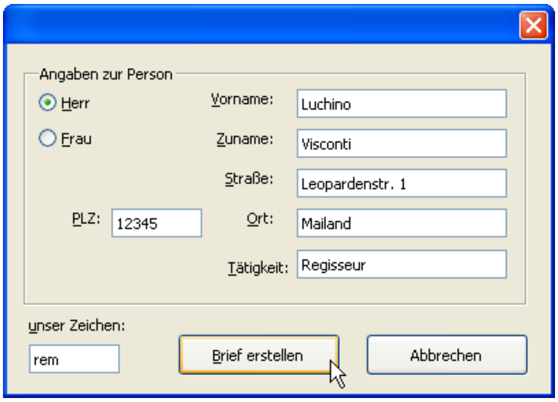


Abbildung 6.125 Der Dialog

Über die Schaltfläche „Brief erstellen“ wird eine Prozedur aufgerufen, in der Variablen für das Dokument, den Cursor, aber auch den Zeilenabstand und die Tabstopps deklariert werden. Die Texte werden als Konstanten hinterlegt (*Beispiele04.sxw*):

```
Sub NeuerBrief_Click

   Dim objDatei As Object
   Dim objCursor As Object
   Dim objStyleFamilies As Object
   Dim objAbsatzvorlagen As Object
   Dim objAbsatzvorlage As Object
   Dim docArgs() As New com.sun.star.beans.PropertyValue
   Dim objZeilenAbstand As New com.sun.star.style.LineSpacing
   Dim objTab(0) As New com.sun.star.style.TabStop
   Dim strUrl As String
   Dim strAnrede As String

   Const FA1 As String = "compurem"
   Const FA2 As String = "Postfach 1299, 80337 München, Germany"
   Const T1 As String = "An den Hausarzt von "
   Const T2 As String = "Die Firma compurem, Werk Entenhausen, " & _
   beabsichtigt, "
   [...]
   Const T8 As String = "Die Kosten für die Untersuchung trägt die " & _
      "Firma compurem, Werk Entenhausen. Eventuell zusätzlich " & _
      "erforderliche Untersuchungen aus gesundheitlichen Gründen " & _
      "sind Kassenleistungen. "

   Const ABSATZ As Integer = 0
```

Aus den Optionsfeldern wird die Anrede ermittelt, die in einer Variablen gespeichert wird:

```
   If Dlg.getControl("optMaennlich").State = True Then
      strAnrede = "Herrn "
   Else
```

```
      strAnrede = "Frau "
   End If
```

Ein neues Dokument wird erzeugt:

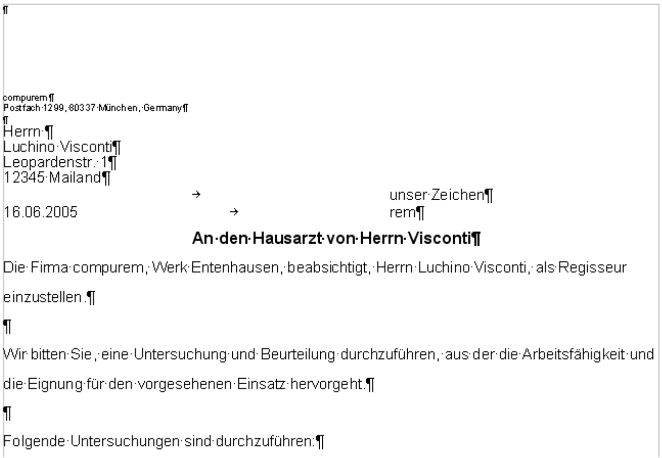
```
   strUrl = "private:factory/swriter"
   objDatei = StarDesktop.loadComponentFromURL _
      (strUrl, "_blank", 0, docArgs())
   objCursor = objDatei.Text.createTextCursor
```

Ein Zeilenabstand und der Abstand für die Tabstops werden festgelegt:

```
   objZeilenAbstand.Mode = com.sun.star.style.LineSpacingMode.FIX
   objZeilenAbstand.Height = 750

   objTab(0).Position = 10000
   objTab(0).Alignment = com.sun.star.style.TabAlign.LEFT
```

Die neuen Vorlagen werden erzeugt:

```
   objStyleFamilies = objDatei.StyleFamilies
   objAbsatzvorlagen = objStyleFamilies.getByName("ParagraphStyles")

   If objAbsatzvorlagen.hasByName("Firmenadresse") = False Then
      objAbsatzvorlage = _
         objDatei.createInstance("com.sun.star.style.ParagraphStyle")
      objAbsatzvorlage.setname("Firmenadresse")
      objAbsatzvorlagen.insertByName("Firmenadresse", objAbsatzvorlage)
   End If
   If objAbsatzvorlagen.hasByName("unserZeichen") = False Then
      objAbsatzvorlage = _
         objDatei.createInstance("com.sun.star.style.ParagraphStyle")
      objAbsatzvorlage.setname("unserZeichen")
      objAbsatzvorlagen.insertByName("unserZeichen", objAbsatzvorlage)
   End If
[...]
```



compurem
Postfach 1299, 80337 München, Germany

Herrn
Luchino Visconti
Leopardenstr. 1
12345 Mailand

unser Zeichen
16.06.2005 rem

An den Hausarzt von Herrn Visconti

Die Firma compurem, Werk Entenhausen, beabsichtigt, Herrn Luchino Visconti, als Regisseur einzustellen.

Wir bitten Sie, eine Untersuchung und Beurteilung durchzuführen, aus der die Arbeitsfähigkeit und die Eignung für den vorgesehenen Einsatz hervorgeht.

Folgende Untersuchungen sind durchzuführen:

Abbildung 6.126 Der erste Teil des Briefes

Den Vorlagen werden die Zeichen- und Absatzinformationen zugewiesen:

```
With objAbsatzvorlagen.getByName("Fließtext")
   .CharFontName = "Arial"
   .CharWeight = com.sun.star.awt.FontWeight.NORMAL
   .CharHeight = 11
   .ParaLineSpacing = objZeilenAbstand
   .ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
End With
With objAbsatzvorlagen.getByName("übi")
   .CharFontName = "Arial"
   .CharWeight = com.sun.star.awt.FontWeight.BOLD
   .CharHeight = 12
   .ParaLineSpacing = objZeilenAbstand
   .ParaAdjust = com.sun.star.style.ParagraphAdjust.CENTER
End With
With objAbsatzvorlagen.getByName("unserZeichen")
   .CharFontName = "Arial"
   .CharWeight = com.sun.star.awt.FontWeight.NORMAL
```

```
    .CharHeight = 11
    .ParaTabStops = objTab()
    .ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
End With
```

Das Objekt „objZeilenabstand“ wird modifiziert und erneut verwendet:

```
With objAbsatzvorlagen.getByName("MA_Adresse")
    .CharFontName = "Arial"
    .CharWeight = com.sun.star.awt.FontWeight.NORMAL
    .CharHeight = 11
    objZeilenAbstand.Height = 400
    .ParaLineSpacing = objZeilenAbstand
    .ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
End With
With objAbsatzvorlagen.getByName("Firmenadresse")
    .CharFontName = "Arial"
    .CharWeight = com.sun.star.awt.FontWeight.NORMAL
    .CharHeight = 7
    .ParaBottomMargin = 2000
    .ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
End With
With objAbsatzvorlagen.getByName("Einzug")
    .CharFontName = "Arial"
    .CharWeight = com.sun.star.awt.FontWeight.NORMAL
    .CharHeight = 11
    .ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
    .ParaLeftMargin = 1000
End With
```

Nun kann der Text eingefügt werden:

```
objCursor.ParaStyleName = "Firmenadresse"
objCursor.gotoEnd(False)
objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
```

Dabei können – falls nötig – Formate der Stile überschrieben werden (wie hier bei „ParaBottom-Margin“):

```
objCursor.String = FA1
objCursor.ParaStyleName = "Firmenadresse"
objCursor.ParaBottomMargin = 0
objCursor.gotoEnd(False)
objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = FA2
objCursor.ParaStyleName = "Firmenadresse"
objCursor.ParaBottomMargin = 0
objCursor.gotoEnd(False)
objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = strAnrede
objCursor.ParaStyleName = "MA_Adresse"
objCursor.gotoEnd(False)
objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
```

Analog werden die Informationen aus dem Dialog ausgelesen und in das Dokument geschrieben:

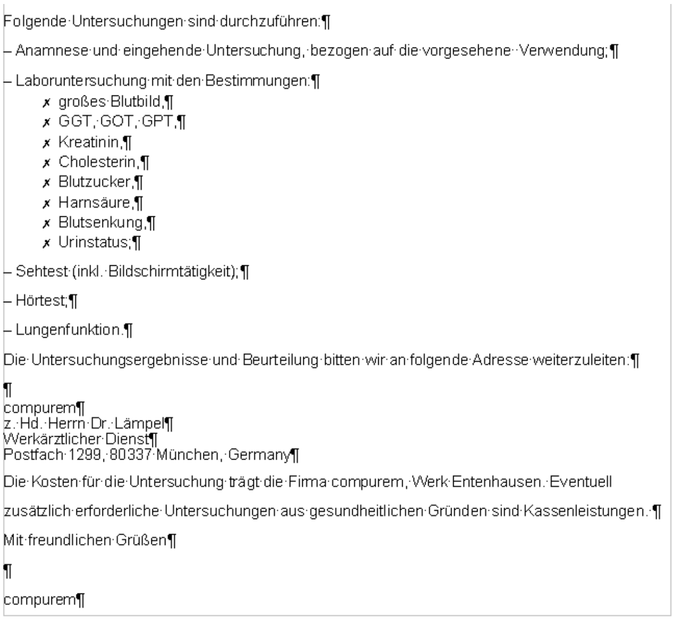
```
objCursor.String = Dlg.getControl("txtVorname").getText() & " " & _
   Dlg.getControl("txtZuname").getText()
objCursor.ParaStyleName = "MA_Adresse"
objCursor.gotoEnd(False)
objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
objCursor.String = Dlg.getControl("txtStrasse").getText()
objCursor.ParaStyleName = "MA_Adresse"
objCursor.gotoEnd(False)
objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
[...]
```

Es können auch vorhandene Stile verwendet werden, wie hier „Aufzählung 2“ und „Aufzählung 5“:

```
objCursor.String = U1
objCursor.NumberingStyleName = "Aufzählung 2"
```

```
   objCursor.gotoEnd(False)
   objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = U2
   objCursor.NumberingStyleName = "Aufzählung 2"
   objCursor.gotoEnd(False)
   objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = U2A
   objCursor.ParaStyleName = "Einzug"
   objCursor.NumberingStyleName = "Aufzählung 5"
   objCursor.gotoEnd(False)
   objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = U2B
   objCursor.NumberingStyleName = "Aufzählung 5"
[...]
   objCursor.gotoEnd(False)
   objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = "Mit freundlichen Grüßen"
   objCursor.ParaStyleName = "Fließtext"
   objCursor.gotoEnd(False)
   objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objDatei.Text.insertControlCharacter(objCursor, ABSATZ, False)
   objCursor.String = FA1
   objCursor.ParaStyleName = "Fließtext"

   Dlg.endExecute()
End Sub
```

Der Dialog wird geschlossen – das Dokument ist erstellt, kann weiter bearbeitet und formatiert werden:



Folgende·Untersuchungen·sind·durchzuführen:¶

– Anamnese·und·eingehende·Untersuchung,·bezogen·auf·die·vorgesehene··Verwendung;¶

– Laboruntersuchung·mit·den·Bestimmungen:¶
- ✗ großes·Blutbild,¶
- ✗ GGT,·GOT,·GPT,¶
- ✗ Kreatinin,¶
- ✗ Cholesterin,¶
- ✗ Blutzucker,¶
- ✗ Harnsäure,¶
- ✗ Blutsenkung,¶
- ✗ Urinstatus;¶

– Sehtest·(inkl.·Bildschirmtätigkeit);¶

– Hörtest;¶

– Lungenfunktion.¶

Die·Untersuchungsergebnisse·und·Beurteilung·bitten·wir·an·folgende·Adresse·weiterzuleiten:¶

¶
compurem¶
z.·Hd.·Herrn·Dr.·Lämpel¶
Werkärztlicher·Dienst¶
Postfach·1299,·80337·München,·Germany¶

Die·Kosten·für·die·Untersuchung·trägt·die·Firma·compurem,·Werk·Entenhausen.·Eventuell
zusätzlich·erforderliche·Untersuchungen·aus·gesundheitlichen·Gründen·sind·Kassenleistungen.·¶

Mit·freundlichen·Grüßen¶

¶

compurem¶

Abbildung 6.127 Der zweite Teil des Briefes

Zugegeben: Dieses Verfahren ist recht umständlich und sollte in dieser Form nicht verwendet werden. Geschickter ist es, mit einer Dokumentenvorlage zu arbeiten, in der sich bereits der Text und die Formate befinden. In sie wird dann lediglich an den entsprechenden Stellen der Text des Dialogs eingefügt. Das oben stehende Beispiel soll jedoch verdeutlichen, wie es möglich wäre, per Programmierung ein Dokument mit Text und komplexen Formaten zu erzeugen.

6.11 Suchen und Ersetzen

Zwar steht Ihnen mit den Cursor-Objekten – den sichtbaren und unsichtbaren – eine ganze Reihe von Möglichkeiten zur Verfügung, jedoch kann es manchmal vorkommen, dass sie für ein effizientes Arbeiten nicht genügen. Für solche Aufgaben bietet sich das Arbeiten mit dem Service com.sun.star.text.TextCursor an.

Zuerst muss ein SearchDescriptor erstellt werden, der festlegt, nach was gesucht wird. Der SearchDescriptor ist ein Objekt, das den Service com.sun.star.util.SearchDescriptor unterstützt und sich über die Methode `createSearchDescriptor` eines Dokumentes erzeugen lässt:

```
Dim objSearchDesc As Object
objSearchDesc = objDoc.createSearchDescriptor
```

Nachdem der SearchDescriptor erzeugt ist, nimmt er den zu suchenden Text entgegen:

```
objSearchDesc.searchString="any text"
```

Über den SearchDescriptor können – analog zum Suchen-Dialog von OOo – die Suchoptionen eingestellt werden. Sie lauten:

Tabelle 6.106 Die Suchoptionen des SearchDescriptors

Eigenschaft	Wert	Beschreibung
SearchString	String	die zu suchende Zeichenkette
ReplaceString	String	die zu ersetzende Zeichenkette
SearchAll	Boolean	ersetze alle
SearchBackwards	Boolean	suche „rückwärts"
SearchCaseSensitive	Boolean	exakte Suche (beachtet Groß- und Kleinschreibung)
SearchRegularExpression	Boolean	sucht reguläre Ausdrücke
SearchSimilarity	Boolean	Ähnlichkeitssuche
SearchSimilarityAdd	Integer	Zeichen hinzufügen bei der Ähnlichkeitssuche
SearchSimilarityExchange		Zeichen tauschen bei der Ähnlichkeitssuche
SearchSimilarityRemove	Integer	Zeichen entfernen bei der Ähnlichkeitssuche
SearchSimilarityRelax	Boolean	Kombinieren bei der Ähnlichkeitssuche
SearchStyles	Boolean	Suche nach Vorlage
SearchWords	Boolean	suche nur ganze Wörter
PropertySetInfo	Object	suche Formate
ValueSearch	Boolean	nur in Selektion
SearchAttributes	Array	zu suchende Attribute
ReplaceAttributes	Array	zu ersetzende Attribute

6.11.1 Die Suche

Ist der SearchDescriptor definiert, kann er auf das Textdokument angewendet werden. Hierzu stellen die OOo-Dokumente die Methoden `findFirst` sowie `findNext` zur Verfügung (*Beispiele06.sxw*):

```
objGefunden = objDoc.FindFirst(objSearchDesc)
Do Until IsNull(objGefunden)
```

```
   ' bearbeite das Suchergebnis
   objGefunden = objDoc.findNext(objGefunden.End, objSearchDesc)
Loop
```

Das Beispiel ermittelt in einer Schleife sämtliche Treffer und liefert ein TextRange-Objekt zurück, das auf die gefundene Textstelle verweist. So könnte man die „Liebe“ fett machen:

```
Dim objSearchDesc As Object
Dim objDoc As Object
Dim objGefunden As Object

objDoc = ThisComponent
objSearchDesc = objDoc.createSearchDescriptor
objSearchDesc.searchString="Liebe"
objGefunden = objDoc.FindFirst(objSearchDesc)
Do Until IsNull(objGefunden)
   objGefunden.CharWeight = com.sun.star.awt.FontWeight.BOLD
   objGefunden = objDoc.findNext(objGefunden.End, objSearchDesc)
Loop
```

II¶

Zielsetzung der folgenden Studien ist es, der Forschungsliteratur zur Geschichte des deutschen Romans eine Monographie zur Bedeutung, Darstellung und Bewertung des Erotischen hinzuzufügen. Angesichts der Tatsache, dass der Roman vom Barock bis hin zur Romantik wesentlich als „**Liebe**sgeschichte" definiert und verfasst ist, bedeutet es in der Tat einen schwerwiegenden Mangel, dass die Anfänge der neueren deutschen Romangeschichte noch nicht unter diesem Gesichtspunkt systematisch dargestellt wurden. Als einzige thematisch-übergreifende Darstellung liegt aus den dreißiger Jahren die Arbeit von W. Stern über den Roman des 17. Jahrhunderts vor: eine außerordentlich gelehrte und materialreiche Studie, welche die Aufnahme und Verarbeitung **liebe**stheoretischer und moralistischer Literatur bei den Autoren des Barock - vor allem des Spätbarock nachweist und darüber hinaus interessante Hinweise auf die Kontinuität bis ins 18. Jahrhundert gibt. Doch dies geschieht in einer eher lexikographischen Manier; dabei gerät das ethische Wirkungsprogramm der Romane außer acht, und Roman- als Kunstgeschichte wird aufgelöst in Begriffs- und Motivgeschichte. Hier wird, wie in den meisten Studien zur '**Liebe**' bei den einzelnen Autoren zu beobachten, über '**Liebe** im Roman', nicht über 'Roman als **Liebe**sgeschichte' oder über '**Liebe**skonzeption des Romans' geschrieben. Die vorliegende Arbeit sucht dies zu revidieren: der Roman wird als komplexe poetische Organisation von 'Welt' und als jeweils exemplarische Verwirklichung desjenigen sujets behandelt, das für die Dichtung im „romantisch-christlichen Zeitalter" (Hegel) das paradigmatische Objekt und das Gleichnis für darzustellendes Leben überhaupt ist; jenes sujet, das der Romanreflexion und der Poetik von Descartes, Leibniz, Opitz, Harsdörffer und Lohenstein bis hin zur Romantik als das 'Romanhafte' schlechthin gilt: für Theologie, Ethik und Poetik ist seit Zesens Grundlegung eines neuen deutschen Romans die „irdische Aphrodite" (Zesen) gehalt- und formkonstitutives Problem der Gattung.¶

Nicht '**Liebe** im Roman', sondern der Roman als Reflexion und poetische Organisation von '**Liebe**' ist also darzustellen.¶

Abbildung 6.128 „Liebe" wird gesucht und gefunden.

6.11.2 Die Ähnlichkeitssuche

Dieses Beispiel zeigt, wie man einen Text nach dem Wort „Roman" durchsucht und die gefundenen Stellen doppelt unterstrichen formatiert. Um neben dem Wort „Roman" auch die Pluralform „Romane" und Deklinationen wie „Romans" zu finden, kommt eine Ähnlichkeitssuche zum Einsatz. Die gefundenen Begriffe dürfen gegenüber dem Suchbegriff bis zu einem Buchstaben mehr oder weniger enthalten und zusätzlich um einen Buchstaben vom Original abweichen:

```
Sub Suchen2
   Dim objSearchDesc As Object
   Dim objDoc As Object
   Dim objGefunden As Object
   objDoc = ThisComponent
   objSearchDesc = objDoc.createSearchDescriptor
```

```
    With objSearchDesc
        .searchString="Roman"
        .SearchSimilarity = True
        .SearchSimilarityAdd = 1
        .SearchSimilarityExchange = 1
        .SearchSimilarityRemove = 1
        .SearchSimilarityRelax = False
    End With
    objGefunden = objDoc.FindFirst(objSearchDesc)
    Do Until IsNull(objGefunden)
        objGefunden.CharUnderline = com.sun.star.awt.FontUnderline.DOUBLE
        objGefunden = objDoc.findNext(objGefunden.End, objSearchDesc)
    Loop
End Sub
```

Der deutsche Roman ist seit Zesen als Liebesgeschichte verfasst. Das bedeutet, er ist wesentlich Pathographie erotischer Selbstverhältnisse und Intersubjektivität. Bereits dies hat ihm allerdings zunächst außerhalb der normativen Poetik, deren Regeln er doch befolgt - die besondere Aufmerksamkeit der Repräsentanten und Sprecher der moralischen Diskurse zugezogen. Die theologischen und philosophischen Kritiker ebenso wie die Rezensenten und Vorredner machen dabei vor allem die spezifische Gesellschaftsferne und „Policeywidrigkeit" (Schlegel der Romanrezeption geltend. Erkannt wird die spezifische Intimität der Romanlektüre und -aufnahme; die Romane gewinnen daraus die Legitimität zur - im Vergleich zu den dramatischen Gattungen - intensiveren Ausleuchtung des Privaten, und dies kann unter dem Primat der Sittenlehre nur bedeuten: zur „censura morum" im Intimen und Subjektiven. Der deutsche Roman seit Zesen ist deshalb im Folgenden zu interpretieren als exquisites und exzessiv beanspruchtes Medium der Einschreibung der sittlichen Institutionen in die Individuen, die zu Subjekten im religiösen, identitätspsychologischen und politischstaatsgesellschaftlichen Sinn werden sollen. Nun hat vor allem M. Foucault - besonders in „Sexualität und Wahrheit" versucht, die Geschichte der Einschreibungen moralisch-politischer Diskurse ins Individuelle nach Seele und Körper zu schreiben. Der deutsche Roman seit Zesen ist in der Tat ars amandi als remedium amoris (Ovid); er kann ergebnisreich gelesen werden als „Geständnis des Fleisches", historisch zunehmend als Kunst im „Gebrauch der Lüste". Unter Berücksichtigung der spezifischen Gattungsregeln und -formen hat sich eine Darstellung der Liebeskonzeption des Romans , welche die Lust-Straf-Schematik in allen Untergattungen zur Kenntnis nimmt, auf Foucaults Fragestellungen zu beziehen. Dies geschieht im Folgenden vor allem dort, wo die romanspezifische Verwirklichung der Glücksstur zschematik als Behandlungsweise erotischer Affekte oder erotisch bestimmter Interpersonalität zur Debatte steht.¶

Abbildung 6.129 „Roman?" wird gekennzeichnet.

6.11.3 Ersetzen von Textteilen

Analog zur Suchfunktion steht die Ersetzungsfunktion von OOo in OOo-Basic zur Verfügung. Die Handhabung der beiden Funktionen ist identisch. Auch für einen Ersetzungsvorgang bedarf es zunächst eines speziellen Objektes, das die Rahmenbedingungen für den Ersetzungsvorgang aufnimmt. Es wird als ReplaceDescriptor bezeichnet und unterstützt den Service com.sun.star.util.ReplaceDescriptor. Sämtliche der oben beschriebenen Eigenschaften des SearchDescriptor werden auch von ReplaceDescriptor unterstützt. So ist es beispielsweise möglich, bei einem Ersetzungsvorgang die Beachtung von Groß- und Kleinschreibung ein- und auszuschalten und Ähnlichkeitssuchen durchzuführen.

Das nachfolgende Beispiel (*Beispiele01.sxw* und *Beispiele01.sxw*) demonstriert den Einsatz eines ReplaceDescriptors für eine Suche innerhalb eines OOo-Dokumentes (siehe Abbildung 6.3).

```
Sub ErsetzeInHTML
   Dim objErsetzen As Object
   Dim objDoc As Object
   Dim objGefunden As Object
   Dim i As Integer
   Dim strTextZeichen(6) As String
   Dim strHTMLZeichen(6) As String

   strTextZeichen() = Array("ä", "Ä", "ö", "Ö", "ü", "Ü", "ß")
   strHTMLZeichen() = Array("&auml", "&Auml;", "&ouml;", "&Ouml;", _
      "&uuml;", "&Uuml;", "&szlig;")
   objDoc = ThisComponent
   objErsetzen = objDoc.createReplaceDescriptor
   objErsetzen.SearchCaseSensitive = True
   For i = 0 To 6
      objErsetzen.SearchString = strTextZeichen(i)
      objErsetzen.ReplaceString = strHTMLZeichen(i)
      objDoc.replaceAll(objErsetzen)
   Next

End Sub
```

Die Begriffe zum Suchen und Ersetzen werden über die Eigenschaften `SearchString` beziehungsweise `ReplaceString` des „ReplaceDescriptors“ gesetzt. Schließlich folgt der eigentliche Ersetzungsvorgang über die Methode `replaceAll` des Dokument-Objektes, die sämtliche Vorkommen des Suchbegriffs ersetzt. Diese Funktion kann verwendet werden, wenn Sie häufig Texte erhalten, die Sie auf die gleiche Art „putzen“ müssen. Das folgende Beispiel löscht alle doppelt vorkommenden Leerzeichen im Dokument:

```
Sub Putzen1

   Dim objDoc As Object
   Dim objErsetzen As Object
   Dim objSuchen As Object

   objDoc = ThisComponent
   objErsetzen = objDoc.createReplaceDescriptor
   objSuchen = objDoc.createSearchDescriptor
   objSuchen.SearchString = "  "
   Do
      With objErsetzen
         .SearchString = "  "
         .ReplaceString = " "
      End With
      objDoc.replaceAll(objErsetzen)
   Loop Until IsNull(objDoc.FindFirst(objSuchen))

End Sub
```

6.11.4 Suchen und Ersetzen mit regulären Ausdrücken

Besonders leistungsfähig ist die Ersetzungsfunktion von OOo im Zusammenhang mit regulären Ausdrücken, die auch in vielen anderen Programmiersprachen (Pearl, PHP, .NET, Java … vorkommen). Diese bieten eine Möglichkeit, anstelle eines festen Wertes einen variablen Suchausdruck mit Platzhaltern und Sonderzeichen zu definieren. Die Liste finden Sie in der Hilfe von OOo.

Tabelle 6.107 Liste der regulären Ausdrücke

Zeichen	**Wirkung/Einsatz**
.	Steht für ein beliebiges einzelnes Zeichen. Der Suchbegriff "Schmi.t" liefert sowohl "Schmitt" als auch "Schmidt".
^Peter	Findet das Wort nur, wenn es am Absatzanfang steht. Sonderinhalte (z. B. Leerfelder und an Zeichen verankerte Rahmen) am Absatzanfang werden ignoriert.
Peter$	Findet das Wort nur, wenn es am Absatzende steht. Sonderinhalte (z. B. Leerfelder und an Zeichen verankerte Rahmen) am Absatzende werden ignoriert.
*	Stellt beliebig viele Zeichen dar. So liefert etwa der Suchbegriff "Ab*c" die Einträge "Ac", "Abc", "Abbc", "Abbbc" usw.
+	Das Zeichen davor muss mindestens einmal bis beliebig oft vorkommen: AX.+4 findet AX 4, aber nicht AX4. Es wird stets der längste mögliche Text innerhalb eines Absatzes gefunden. Wenn der Absatz den Text AX 4 AX4 enthält, wird er vom ersten A bis zur letzten 4 ausgegeben.
?	Findet Wörter, in denen der Buchstabe vor diesem Zeichen nicht unbedingt vorkommen muss. So findet z. B. "Texte?" sowohl "Texte" als auch "Text".
\C	Findet genau dieses angegebene Zeichen (keine Ziffer), hier C (wenn Sie z. B. in einem regulären Ausdruck nach dem Dollarzeichen selbst suchen wollen: \$)
\n	Findet einen mit Umschalt+Eingabetaste eingefügten Zeilenumbruch. Indem Sie \n in die Felder Suchen nach und Ersetzen durch eingeben und auf Ersetze alle klicken, tauschen Sie Zeilenumbrüche gegen Absatzumbrüche aus.
\t	Findet einen Tabulator. Dieser Ausdruck kann auch im Feld Ersetzen durch verwendet werden.
\>	Findet das Wort nur, wenn es am Wortende steht. Zum Beispiel liefert der Suchbegriff "Buch\>" das Wort "Lesebuch", nicht aber "Buchdeckel".
\<	Findet das Wort nur, wenn es am Wortanfang steht. Zum Beispiel liefert der Suchbegriff "\>Buch" das Wort "Buchdeckel", nicht aber "Lesebuch".
^$	Findet leere Absätze.
^.	Sucht das erste Zeichen eines Absatzes.
&	Die anschließend an diesen Ausdruck in das Feld Ersetzen durch eingegebenen Zeichen werden bei einer Ersetzung zum Suchbegriff hinzugefügt. Wenn Sie in Suchen nach zum Beispiel "Fenster" und in Ersetzen durch "&rahmen" eingeben, wird das Wort "Fenster" durch "Fensterrahmen" ersetzt.

<table>
<tr><th>Zeichen</th><th>Wirkung/Einsatz</th></tr>
<tr><td></td><td>Außerdem können Sie mit dem Ausdruck "&" im Feld Ersetzen durch die Attribute oder das Format des in Suchen nach angegebenen Suchbegriffs ändern.</td></tr>
<tr><td>[abc123]</td><td>Findet die Zeichen in der Klammer.</td></tr>
<tr><td>[a-e]</td><td>Findet die Zeichen zwischen a und e.</td></tr>
<tr><td>[a-eh-x]</td><td>Findet die Zeichen zwischen a-e und h-x.</td></tr>
<tr><td>[^a-s]</td><td>Findet alle Zeichen außer a-s</td></tr>
<tr><td>\xXXXX</td><td>Findet ein Zeichen auf Grundlage seines vierstelligen Hexadezimalcodes (XXXX).
Der Code des Zeichens hängt von der jeweiligen Schrift ab. Die Codes können Sie unter Einfügen | Sonderzeichen einsehen.</td></tr>
<tr><td>dies|das</td><td>Findet die Wörter vor und hinter dem "|". Beispielsweise findet "dies|das" sowohl "dies" als auch "das".</td></tr>
<tr><td>{2}</td><td>Gibt an, wie oft das Zeichen vor der öffnenden Klammer im Wort vorkommen muss. Zum Beispiel liefert der Suchbegriff "Man{2}" das Wort "Mann".</td></tr>
<tr><td>{1,2}</td><td>Gibt an, wie oft das Zeichen vor der öffnenden Klammer im Wort vorkommen darf. Zum Beispiel liefert der Suchbegriff "Man{1,2}" sowohl "Mann" als auch "man".</td></tr>
<tr><td>( )</td><td>Die in der Klammer enthaltenen Zeichen gelten als Referenz. Auf die erste Referenz im aktuellen Ausdruck können Sie dann mit "\1" Bezug nehmen, auf die zweite mit "\2" usw.
Enthält Ihr Text zum Beispiel die Zahl 13487889 und Sie führen eine Suche mit dem Ausdruck (8)7\1\1 durch, dann wird "8788" gefunden.</td></tr>
<tr><td>[:digit:]?</td><td>Findet eine einstellige Zahl. Für die Suche nach mehrstelligen Zahlen verwenden Sie [:digit:]*.</td></tr>
<tr><td>[:space:]?</td><td>Findet Leerraum wie Leerzeichen und Tabulator.</td></tr>
<tr><td>[:print:]?</td><td>Findet druckbares Zeichen.</td></tr>
<tr><td>[:cntrl:]?</td><td>Findet Steuerzeichen.</td></tr>
<tr><td>[:alnum:]?</td><td>Findet Buchstaben und Zahlen (alphanumerische Zeichen).</td></tr>
<tr><td>[:alpha:]?</td><td>Findet Buchstaben.</td></tr>
<tr><td>[:lower:]?</td><td>Findet Kleinbuchstaben, sofern unter Optionen das Feld Exakte Suche markiert ist.</td></tr>
<tr><td>[:upper:]?</td><td>Findet Großbuchstaben, sofern unter Optionen das Feld Exakte Suche markiert ist.</td></tr>
</table>

Das folgende Beispiel (*Beispiele07.sxw*) zeigt, wie sich mit Hilfe des regulären Ausdrucks $ sämtliche Absatzzeichen aus einem Textdokument entfernen lassen und durch ein Leerzeichen ersetzen, damit nicht aufeinander folgende Wörter ohne Leerschritt nebeneinander stehen:

```
Sub ErsetzenDurchRegulaereAusdruecke
   Dim objDoc As Object
   Dim objErsetzen As Object
   objDoc = ThisComponent
   objErsetzen = objDoc.createReplaceDescriptor
   With objErsetzen
      .SearchRegularExpression = True
      .SearchString = "$"
      .ReplaceString = " "
   End With
   objDoc.replaceAll(objErsetzen)
End Sub
```

Messies¶

„‚Messie'·kommt·von·dem·englischen·Wort·‚mess',·also·¶

Unordnung,·Wirrwarr,·Chaos·und·Schwierigkeiten.·¶

Messies·bewahren·beispielsweise·für·den·Fall·aller·¶

Fälle·alles·Mögliche·auf.·Sie·haben·Schwierigkeiten·aufzuräumen·und·leben·ständig·im·Chaos.·Das Schlimme·ist,·dass·es·keiner·merkt.·Messies·schämen·sich·oft.·Normalerweise·wird·kaum·ein Mensch·seine·Wohnung·zu·sehen·bekommen.·Deshalb·sind·Messies·auch·nahezu·unsichtbar·für·die meisten·Mitbürger.·Denn·nach·außen·funktionieren·sie·ja.¶

Lösungsansätze:¶

Der·Messie·muss·von·sich·aus·aktiv·werden!·Manchmal·können·Außenstehende·helfen,·wenn·sie·es schaffen,·sich·in·die·Problematik·des·Messie·einzudenken.·Aber·es·ist·schwierig,·für·so·ein·Leiden Verständnis·entwickeln·zu·können.·Man·könnte·nicht·einfach·Hilfe·anbieten,·ohne·Vorschriften·zu machen.·Für·dieses·Phänomen·gibt·es·keine·Patentrezepte!"·¶

Messies·„‚Messie'·kommt·von·dem·englischen·Wort·‚mess',·also··Unordnung,·Wirrwarr,·Chaos·und Schwierigkeiten.··Messies·bewahren·beispielsweise·für·den·Fall·aller··Fälle·alles·Mögliche·auf.·Sie haben·Schwierigkeiten·aufzuräumen·und·leben·ständig·im·Chaos.·Das·Schlimme·ist,·dass·es·keiner merkt.·Messies·schämen·sich·oft.·Normalerweise·wird·kaum·ein·Mensch·seine·Wohnung·zu·sehen bekommen.·Deshalb·sind·Messies·auch·nahezu·unsichtbar·für·die·meisten·Mitbürger.·Denn·nach außen·funktionieren·sie·ja.·Lösungsansätze:·Der·Messie·muss·von·sich·aus·aktiv·werden!·Manchmal können·Außenstehende·helfen,·wenn·sie·es·schaffen,·sich·in·die·Problematik·des·Messie einzudenken.·Aber·es·ist·schwierig,·für·so·ein·Leiden·Verständnis·entwickeln·zu·können.·Man könnte·nicht·einfach·Hilfe·anbieten,·ohne·Vorschriften·zu·machen.·Für·dieses·Phänomen·gibt·es keine·Patentrezepte!"·¶

Abbildung 6.130 Vorher und nachher

Ebenso kann mit „^$" nach leeren Absätzen gesucht werden (*Beispiele07.sxw*). Oder Leerzeichen zu Beginn eines Absatzes können entfernt werden:

```
Sub Putzen2

   Dim objDoc As Object
   Dim objErsetzen As Object
```

```
    objDoc = ThisComponent
    objErsetzen = objDoc.createReplaceDescriptor
    objSuchen = objDoc.createSearchDescriptor
    With objErsetzen
       .SearchRegularExpression = True
       .SearchString = "^ "
       .ReplaceString = ""
    End With
    objDoc.replaceAll(objErsetzen)
End Sub
```

6.12 Weitere Elemente außer Text

Bislang wurde in diesem Kapitel nur Text besprochen: Text erzeugen, Text formatieren und Text auslesen. Jedoch kann ein Writer-Dokument auch über andere Elemente verfügen: Tabellen, Grafiken, Felder, Frames, Linien, Querverweise, Textmarken …

Diese Objekte können an jeder beliebigen Position innerhalb eines Textes verankert werden. Aufgrund dieser Gemeinsamkeiten unterstützen alle diese Objekte in OOo einen gemeinsamen Basis-Service namens com.sun.star.text.TextContent. Er stellt folgende Eigenschaften zur Verfügung:

- AnchorType (Enum) – bestimmt den Verankerungstyp eines TextContent-Objekts (Vorgabewerte gemäß Aufzählung com.sun.star.text.TextContentAnchorType).
- AnchorTypes (sequence of Enum) – Aufzählung aller AnchorTypes, die ein spezielles TextContent-Objekt unterstützt.
- TextWrap (Enum) – bestimmt den Umlauftyp um ein TextContent-Objekt (Vorgabewerte gemäß Aufzählung com.sun.star.text.WrapTextMode).

Auch einige Methoden teilen sich die TextContent-Objekte – insbesondere die zum Erzeugen, Einfügen und Löschen von Objekten.

- Das Erzeugen eines neuen TextContent-Objektes erfolgt über die Methode `createInstance` des Dokument-Objektes.
- Das Einfügen eines Objektes erfolgt über die Methode `insertTextContent` des Text- Objektes.
- Das Löschen von TextContent-Objekten erfolgt über die Methode `removeTextContent`.

In den nachfolgenden Abschnitten findet sich eine Reihe von Beispielen, die diese Methoden verwenden.

6.13 Tabellen

6.13.1 Tabellen einfügen

Das nachfolgende Beispiel (*Beispiele08.sxw*) erzeugt eine Tabelle unter Zuhilfenahme der weiter oben beschriebenen Methode `createInstance`.

```
Sub Tabellen1

   Dim objDoc As Object
   Dim objTabelle As Object
   Dim objCursor As Object
   Dim strURL As String
   Dim docArgs()

   strUrl = "private:factory/swriter"
   objDoc = StarDesktop.loadComponentFromURL _
      (strUrl, "_blank", 0, docArgs())
   objCursor = objDoc.Text.createTextCursor()
   objTabelle = objDoc.createInstance("com.sun.star.text.TextTable")
   objTabelle.initialize(5, 4)
   objDoc.Text.insertTextContent(objCursor, objTabelle, False)

End Sub
```

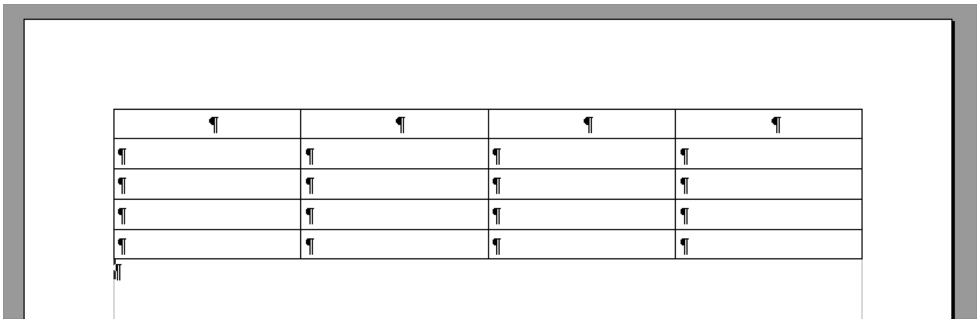


Abbildung 6.131 Die neue Tabelle

Nach der Erzeugung wird die Tabelle über die Methode `initialize` auf die gewünschte Anzahl der Zeilen und Spalten eingestellt und dann über `insertTextContent` in das Textdokument eingefügt.

Die Methode `insertTextContent` verlangt neben dem einzufügenden Content-Objekt noch zwei weitere Parameter:

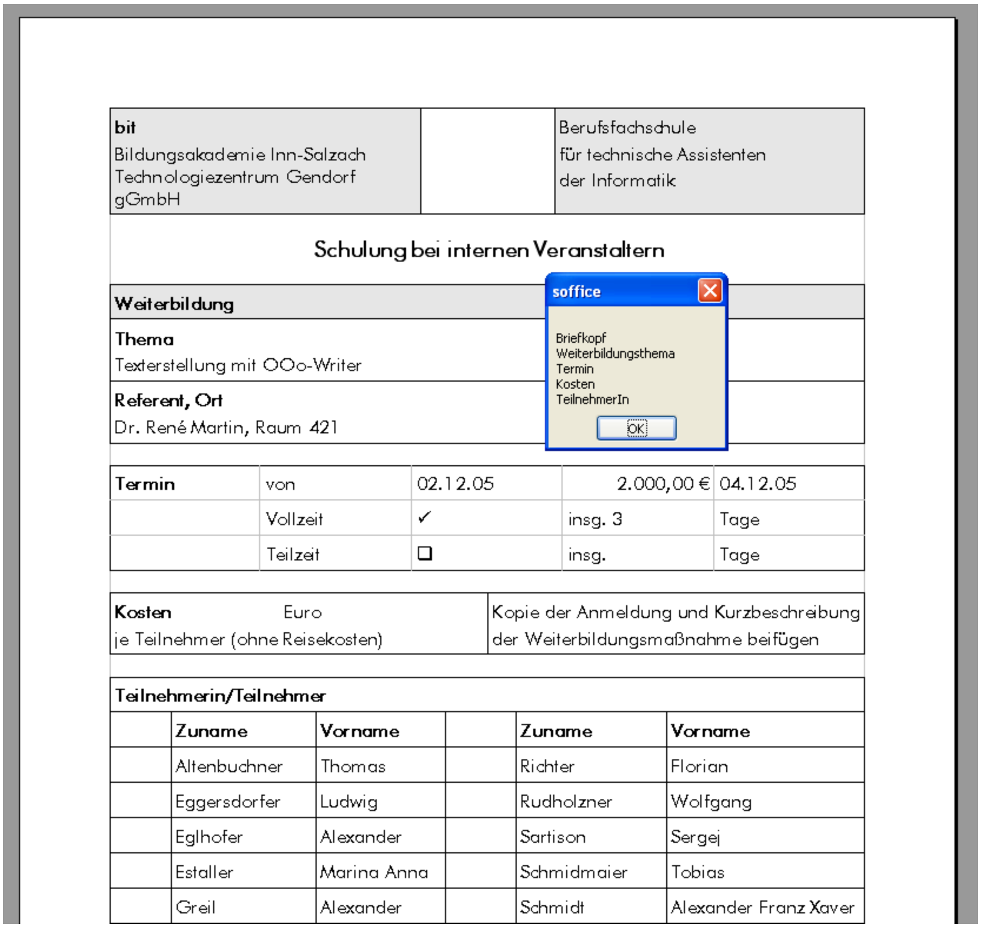
- ein Cursor-Objekt, das die Einfügeposition bestimmt
- eine boolesche Variable, die angibt, ob das Content-Objekt die aktuelle Selektion des Cursors ersetzen (Wert „True“) oder vor dieser in den Text eingefügt werden soll („False“)

6.13.2 Tabellen auslesen

Die in einem Textdokument vorhandenen Tabellen können mit der Methode `getTextTables()` oder mit der Eigenschaft `TextTables` erfasst werden. Ihre Anzahl beträgt `Count` oder `getCount()`. Mit ihrer Hilfe kann ein Dokument über eine Schleife durchlaufen werden und alle Tabellen „namentlich“ einsammeln (*Beispiele08.sxw*):

```
Sub Tabellen2

   Dim objDoc As Object
   Dim objTabellen As Object
   Dim objTabelle As Object
   Dim i As Integer
   Dim strTabellenName As String

   objDoc = ThisComponent
   objTabellen = objDoc.getTextTables()
   For i = 0 to objTabellen.count - 1
      objTabelle = objTabellen(i)
      strTabellenName = strTabellenName & Chr(13) & objTabelle.Name
   Next

MsgBox strTabellenName
```

```
End Sub
```



bit Bildungsakademie Inn-Salzach Technologiezentrum Gendorf gGmbH		Berufsfachschule für technische Assistenten der Informatik

Schulung bei internen Veranstaltern

Weiterbildung
Thema Texterstellung mit OOo-Writer
Referent, Ort Dr. René Martin, Raum 421

Termin	von	02.12.05	2.000,00 €	04.12.05
	Vollzeit	✓	insg. 3	Tage
	Teilzeit	❑	insg.	Tage

Kosten Euro je Teilnehmer (ohne Reisekosten)	Kopie der Anmeldung und Kurzbeschreibung der Weiterbildungsmaßnahme beifügen

Teilnehmerin/Teilnehmer

	Zuname	Vorname		Zuname	Vorname
	Altenbuchner	Thomas		Richter	Florian
	Eggersdorfer	Ludwig		Rudholzner	Wolfgang
	Eglhofer	Alexander		Sartison	Sergej
	Estaller	Marina Anna		Schmidmaier	Tobias
	Greil	Alexander		Schmidt	Alexander Franz Xaver

Abbildung 6.132 Die Tabellen werden namentlich ausgelesen.

Die Methode `hasElements()` überprüft, ob ein Dokument Tabellen besitzt:

```
MsgBox objDoc.getTextTables().hasElements()
```

mit `hasByName(String)` kann überprüft werden, ob eine bestimmte Tabelle im Dokument vorhanden ist, mit `getByName(String)` kann darauf zugegriffen werden.

6.13.3 Tabellen bearbeiten

Die Tabellen in Writer haben sehr viel Ähnlichkeit mit ihren Kollegen aus Calc. Eine Tabelle setzt sich zusammen aus einzelnen Zeilen. Diese wiederum enthalten die verschiedenen Zellen. Streng genommen gibt es in OOo keine Tabellenspalten. Diese ergeben sich implizit durch die gleiche Anordnung der untereinander stehenden Zeilen. Um den Zugriff auf die Tabellen zu vereinfachen, bietet OOo dennoch einige Methoden an, die spaltenweise arbeiten. Sie sind nutzbar, sofern die Tabelle ohne Zellzusammenfassungen auskommt.

Tabelle 6.108 Der Service SwXTextTable stellt folgende Eigenschaften für Tabellen zur Verfügung

1. **Eigenschaft**	1. **Wert**	1. **Beschreibung**
2. **Tabelle**	2.	2.
3. Name	3. String	3. Name der Tabelle
4. IsWidthRelative	4. Boolean	4. relative Breite
5. RelativeWidth	5. Integer	5. Breite
6. LeftMargin, RightMargin, TopMargin, BottomMargin	6. Long	6. linker, rechter, oberer, unterer Abstand in 100stel Millimetern
7. HoriOrient	7. Integer	7. horizontale Ausrichtung des Typs com.sun.star.text.HoriOrientation: 8. NONE (0), RIGHT (1), CENTER (2), LEFT (3), INSIDE (4), OUTSIDE (5), FULL (6), LEFT_AND_WIDTH (7)
8. **Textfluss**	8.	9.
9. BreakType	9. Long	10. Umbruch vor dem Absatz des Typs com.sun.star.style.BreakType: 11. NONE, COLUMN_BEFORE, COLUMN_AFTER, COLUMN_BOTH, PAGE_BEFORE, PAGE_AFTER, PAGE_BOTH
10. Width	10. Long	12. Tabellenbreite (schreibgeschützt)
11. PageDescName	11. String/ void	13. Seitenumbruch (mit Namen)
12. PageNumberOffset	12. Integer/ void	14. Seitennummerierung
13. Split	13. Boolean	15. Tabelle nicht trennen
14. TextWrap	14. Long	16. mit folgendem Absatz zusammenhalten

1. Eigenschaft	1. Wert	1. Beschreibung
15. RepeatHeadline	15. Boolean	17. Überschrift wiederholen
16. Spalten	**16.**	**18.**
17. BottomMargin	17. Long	19. unterer Abstand in 100stel Millimetern
18. KeepTogether	18. Boolean	20. mit folgendem Absatz zusammenhalten
19. Umrandung	**19.**	**21.**
20. TableBorder	20. Array of struct	22. bestimmt die jeweilige Linie nach com.sun.star.table.BorderLine: 23. TopLine, BottomLine, LeftLine, RightLine, HorizontalLine und VerticalLine 24. Distance 25. IsTopLineValid, IsBottomLineValid, IsLeftLineValid, IsRightLineValid, IsHorizontalLineValid, IsVerticallLineValid und IsDistanceValid
21. Hintergrund	**21.**	**26.**
22. BackColor	22. Long	27. Hintergrundfarbe
23. BackGraphicFilter	23. String	28. Filter des Hintergrundbildes
24. BackGraphicLocation	24. Long	29. Eine Konstante der Aufzählung com.sun.star.style.GraphicLocation: 30. NONE, LEFT_TOP, MIDDLE_TOP, RIGHT_TOP, LEFT_MIDDLE, MIDDLE_MIDDLE, RIGHT_MIDDLE, LEFT_BOTTOM, MIDDLE_BOTTOM, RIGHT_BOTTOM, AREA, TILED
25. BackGraphicURL	25. String	31. Speicherort der Grafik
26. BackTransparent	26. Boolean	32. transparenter Hintergrund
27. ShadowFormat	27. Array of struct	33. bestimmt das Linienformat nach com.sun.star.table.ShadowFormat: 34. Location (Konstanten gemäß com.sun.star.table.ShadowLocation: NONE, TOP_LEFT, TOP_RIGHT, BOTTOM_LEFT, BOTTOM_RIGHT) 35. ShadowWidth (integer) 36. IsTransparent (Boolean) 37. Color (Long)

1. **Eigenschaft**	1. **Wert**	1. **Beschreibung**
28. Sonstiges	**28.**	**38.**
29. LinkDisplayName	29. String	39. Beschriftung des Hyperlinks
30. ChartColumnAsLabel/ChartRowAsLabel	30. Boolean	40. die erste Spalte/Zeile dient als Beschriftung, falls die Tabelle für ein Diagramm verwendet wird
31. StartRedline/EndRedline	31. Array/ Void	41.
32. TableColumnRelativeSum	32. Integer	42. Summe
33. TableColumnSeparators	33. Array/ Void	43. Spaltentrennzeichen
34. TextSection	34. Object/ Void	44.
35. WritingMode	35. Integer	45.
36. Rows/Columns	36. Object	46. Zeilen und Spalten
37. CellNames	37. Array	47. Zellnamen
38. AnchorType	38. Long	48. Art des Textankers
39. AnchorTypes	39. Array	49. mehrere Textanker
40. Anchor	40. Objekt	50. Textanker

Tabelle 6.109 Die Methoden des Tabellenobjekts

Methode	**Rückgabewert**	**Beschreibung**
initialize (Long, Long)	Void	setzt die Anzahl der Zeilen und Spalten (siehe die Beispiele oben)
getAnchor (Void)	Object	Textrange, der angibt, wo das Ankerobjekt liegt
getRows (Void)	Object	das Zeilenobjekt
getColumns (Void)	Object	das Spaltenobjekt
getCellByName (String)	Object	gibt Zelle zurück („C5“)
getCellRangeByName (String)	Object	Zelle oder Zellbereich („C5“ oder „C5:D7“)
getCellNames (Void)	Array	die Sammlung der Zellen
createCursorByCellName (String)	Object	erstellt ein Cursorobjekt an der Zellposition
getCellByPosition (Long, Long)	Object	liefert die Zelle nach Spalten

Methode	Rückgabewert	Beschreibung
		und Zeilen
getCellRangeByPosition (Long, Long, Long, Long)	Object	Zelle: links: links, oben, rechts, unten
getNotANumber (Void)	Double	
isNotANumber (Double)	Boolean	
getData (Void)	Array	numerische Daten
setData (Array)	Void	numerische Daten
getDataArray (Void)	Array	numerische Daten oder Text
setDataArray (Array)	Void	numerische Daten oder Text
getRowDescriptions (Void)	Array	Beschreibung der Information der Zeilen
setRowDescriptions (Array)	Void	Beschreibung der Information der Zeilen
getColumnDescriptions (Void)	Array	Beschreibung der Information der Spalten
setColumnDescriptions (Array)	Void	Beschreibung der Information der Spalten
getName (Void)	String	der Name der Tabelle
setName (String)	Void	ändert den Namen der Tabelle
autoFormat (String)	Void	weist das Autoformat zu
createSortDescriptor (Void)	Array	erstellt ein Objekt zum Sortieren der Zeilen
sort (Array)	Void	sortiert

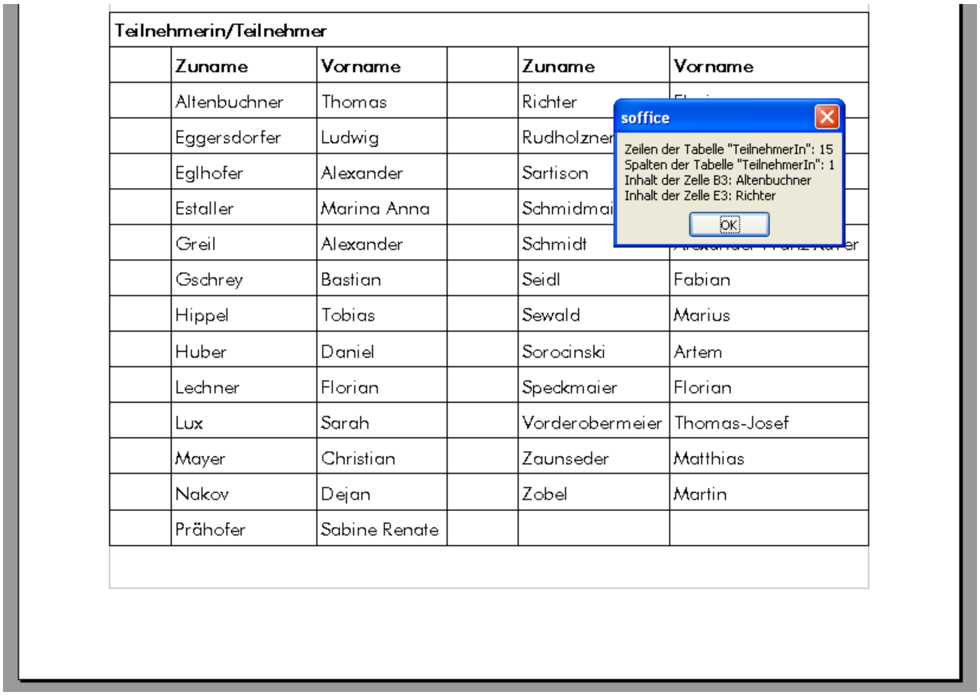
Das Zeilen- und Spaltenobjekt (`getRows()` und `getColumns()`) kann über die Eigenschaft `Count` oder über die Methode `getCount()` gezählt werden. `HasElements()` liefert, ob es Zeilen oder Spalten hat, `getByIndex(Long)` greift auf eine Zeile zu, `insertByIndex(Long, Long)` fügt ein und `removeByIndex(Long, Long)` löscht Zeilen oder Spalten. Die erste der beiden Zahlen gibt dabei jeweils den Index der Position an. Beachten Sie, dass die Zählung bei 0 beginnt!

Die Methode `insertByIndex(Long, Long)` fügt die Zeilen immer oberhalb der Tabelle ein! Es existiert leider keine Methode, um die Zeilen darunter einzufügen.

Diese beiden Methoden können nur verwendet werden, wenn keine Zellen verbunden wurden!

Mit diesen Methoden können Informationen über Tabellen ermittelt werden: die Anzahl der Zeilen, Spalten und Inhalte von Zellen:

```
Sub Tabellen3
   Dim objDoc As Object
   Dim objTabellen As Object
   Dim objTabelle As Object
   Dim objZeilen As Object
   Dim objSpalten As Object
   Dim objZelle As Object
   Dim strTabellenInfo As String
   Dim strAlleZellNamen() As String
   Dim strZellName As String
   Dim i As Integer
   objDoc = ThisComponent
   objTabellen = objDoc.getTextTables()
   If objTabellen.hasByName("TeilnehmerIn") Then
      objTabelle = objTabellen.getByName("TeilnehmerIn")
      objZeilen = objTabelle.getRows()
      objSpalten = objTabelle.getColumns()
      strTabellenInfo = "Zeilen der Tabelle ""TeilnehmerIn"": " & _
         objZeilen.Count
      strTabellenInfo = strTabellenInfo & Chr(13) & _
         "Spalten der Tabelle ""TeilnehmerIn"": " & objSpalten.Count
      objZelle = objTabelle.getCellByName("B3")
      strTabellenInfo = strTabellenInfo & Chr(13) & _
         "Inhalt der Zelle B3: " & objZelle.String
      objZelle = objTabelle.getCellByPosition(4, 2)
      strTabellenInfo = strTabellenInfo & Chr(13) & _
         "Inhalt der Zelle E3: " & objZelle.String
      MsgBox strTabellenInfo
```



Teilnehmerin/Teilnehmer					
	Zuname	Vorname		Zuname	Vorname
	Altenbuchner	Thomas		Richter	[illegible]
	Eggersdorfer	Ludwig		Rudholzner	[illegible]
	Eglhofer	Alexander		Sartison	[illegible]
	Estaller	Marina Anna		Schmidmai	[illegible]
	Greil	Alexander		Schmidt	[illegible]
	Gschrey	Bastian		Seidl	Fabian
	Hippel	Tobias		Sewald	Marius
	Huber	Daniel		Sorocinski	Artem
	Lechner	Florian		Speckmaier	Florian
	Lux	Sarah		Vorderobermeier	Thomas-Josef
	Mayer	Christian		Zaunseder	Matthias
	Nakov	Dejan		Zobel	Martin
	Prähofer	Sabine Renate			

Abbildung 6.133 Die Inhalte werden ausgelesen.

Dabei fallen mehrere Dinge auf: Die Zählung beginnt – wie fast überall in OOo-Basic – bei 0. Das heißt, dass auf die Zelle „B3" über ihren Namen oder über `objTabelle.getCellByPosition(1, 2)` zugegriffen wird. Die Methode `objTabelle.getCellByPosition` verwendet zuerst die Spaltenangabe und dann die Zeileninformation – also wie im Deutschen „A1" und nicht „R1C1".

Übrigens stehen die Namen der Zellen in der Statusleiste am rechten Rand.

Beachten Sie, dass die Methode `initialize` zuerst die Zeilenanzahl benötigt und anschließend die Spaltenzahl – also genau umgekehrt als `getCellByPosition`.

Da die erste Zeile verbunden ist, „erkennt" das Objekt `getColumns()` nur eine Spalte, obwohl die Tabelle ab der zweiten Zeile sechs Spalten aufweist. Der Zugriff über `getCellByPosition` und `getCellByName` funktioniert anstandslos.

Wenn Sie nun von solchen komplexen Tabellen herausfinden müssen, welche Zellen vorhanden sind, dann liefert die Methode `getCellNames()` die Sammlung aller Zellnamen, in unserem Beispiel sieht das wie folgt aus:

```
strAlleZellNamen = objTabelle.getCellNames()
```

```
      For i = 0 To UBound(strAlleZellNamen)
         strZellName = strZellName & " / " & strAlleZellNamen(i)
      Next

      MsgBox "Folgende Zellen sind in der Tabelle vorhanden:" & _
         Chr(13) & strZellName
   End If
End Sub
```

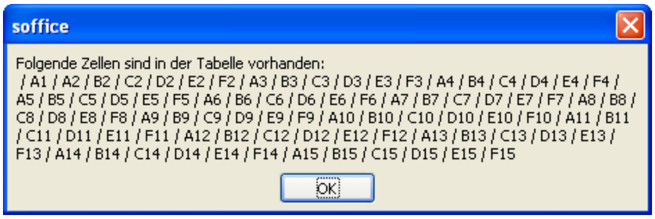


Abbildung 6.134 Die Zellnamen werden ausgelesen.

Das Zeilenobjekt verfügt über folgende Eigenschaften:

BackColor, BackGraphicFilter, BackGraphicLocation, BackGraphicURL, BackTransparent, Height, IsAutoHeight, SizeType und TableColumnSeparators. Sie sind alle in Tabelle 6.15 des Tabellenobjekts beschrieben.

Die Zelle verfügt über folgende Eigenschaften:

BackColor, BackGraphicFilter, BackGraphicLocation, BackGraphicURL, BackTransparent, BorderDistance, BottomBorder, BottomBorderDistance, CellName, EndRedline, IsProtected, LeftBorder, LeftBorderDistance, NumberFormat, RightBorder, RightBorderDistance, StartRedline, TextSection, TopBorder, TopBorderDistance, UserDefinedAttributes, VertOrient, Formula, Value, Type, Error, Text, Start, End, String,

Sie sind weitgehend in Tabelle 6.15 beschrieben.

Neben den Formatierungseigenschaften (BackColor, BottonBorder, und so weiter) erscheinen die Eigenschaften für den Inhalt wichtig:

Tabelle 6.110 Wichtige Eigenschaften für den Zellinhalt

Eigenschaft	Wert	Beschreibung
String	String	der Text
Text	Objekt	ein Textobjekt – keine Zeichenkette!

Eigenschaft	Wert	Beschreibung
Formula	String	ein Rechenausdruck
Value	Double	eine Zahl (mit der gerechnet wird)
Error	Long	eine Fehlermeldung (Nummer) einer Berechnung

Und damit werden auch die folgenden Methoden des Zellobjekts transparent:

Tabelle 6.111 Einige Methoden der Zelle

Methode	Wert	Beschreibung
getFormula (Void)	String	liefert die Formel (oder den Inhalt)
setFormula (STRING)	Void	schreibt eine Formel (oder den Inhalt)
getString (Void)	String	liefert den Inhalt
setString (String)	Void	schreibt einen Inhalt
getValue (Void)	Double	liefert einen Zahlenwert
setValue (Double)	Void	schreibt einen Zahlenwert
getType (Void)	Long	liefert einen Typ der Aufzählung com.sun.star.table.Cell-ContentType: EMPTY, VALUE, TEXT oder FORMULA
getError (Void)	Long	liefert die Fehlernummer

Das folgende Beispiel (*Beispiele08.sxw*) erstellt in einem neuen Dokument eine Tabelle mit zwei Spalten und zwei Zeilen:

```
strUrl = "private:factory/swriter"
objDoc = StarDesktop.loadComponentFromURL _
   (strUrl, "_blank", 0, docArgs())

objTextCursor = objDoc.Text.createTextCursor()
objTabelle = objDoc.createInstance("com.sun.star.text.TextTable")
objTabelle.initialize(2, 2)
objDoc.Text.insertTextContent(objTextCursor, objTabelle, False)
```

Eigentlich sollte man die Linien wegformatieren können, aber hier scheint ein Bug vorzuliegen:

```
objTabelle.TableBorder.TopLine.OuterLineWidth = 0
objTabelle.TableBorder.LeftLine.OuterLineWidth = 0
```

```
objTabelle.TableBorder.BottomLine.OuterLineWidth = 0
objTabelle.TableBorder.RightLine.OuterLineWidth = 0
```

Die Schrift innerhalb der Tabelle wird formatiert:

```
objZellCursor = objTabelle.getCellRangeByName("A1:B2")
With objZellCursor
   .ParaStyleName = "Standard"
   .CharFontName = "Arial"
   .CharHeight = 11
   .CharUnderline = com.sun.star.awt.FontUnderline.NONE
   .CharWeight = com.sun.star.awt.FontWeight.NORMAL
   .CharPosture = com.sun.star.awt.FontSlant.NONE
   .ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
   .ParaLineSpacing = LS
   .ParaBottomMargin = 0
End With
```

Die Tabelle wird etwas kleiner formatiert und zentriert:

```
objTabelle.HoriOrient = com.sun.star.text.HoriOrientation.CENTER
objTabelle.RelativeWidth = 75
```

Die erste Zeile wird grau formatiert:

```
objTabelle.getCellRangeByName("A1:B1").BackColor = RGB(200, 200, 200)
```

Auf die erste Zeile könnte man auch so zugreifen:

```
objTabelle.getRows().getByIndex(0).BackColor = RGB(200, 200, 200)
```

Vom Benutzer werden vier Eingaben verlangt, wobei lediglich die erste notwendig ist – die anderen drei sind optional:

```
Do
   dblZulage = InputBox("Bitte geben Sie einen Betrag für " & _
      "die übertarifliche Zulage ein!")
   If dblZulage = 0 Then MsgBox "Bitte geben Sie eine Zahl ein!"
Loop Until dblZulage <> 0
dblSchichtzulage = InputBox("Bitte geben Sie einen Betrag für " & _
      "die Schichtzulage ein!")
dblErschwernisZulage = InputBox("Bitte geben Sie einen Betrag für" & _
```

```
        " die Erschwerniszulage ein!")
   dblSchweisserzulage = InputBox("Bitte geben Sie einen Betrag für " & _
        "die Schweisserzulage ein!")
```

Zuerst wird die Überschrift eingetragen:

```
   objZelle = objTabelle.getCellByPosition(0, 0)
   objTextCursor = objZelle.createTextCursor()
   objTextCursor.String = "Übertarifliche Zulage"
   objTabelle.getCellByPosition(1, 0).Value = dblZulage
   objTabelle.getCellByPosition(1, 0).NumberFormat = 106
```

Dabei könnte der Text auch ohne Zellcursor direkt in die Zelle geschrieben werden, wie die übrigen drei Beispiele zeigen:

```
If dblSchichtzulage <> 0 Then
   objTabelle.getRows().insertByIndex(objTabelle.getRows().Count -1, 1)
   objTabelle.getCellByPosition(0, objTabelle.getRows().Count -2). _
      String = "Schichtzulage"
   objTabelle.getCellByPosition(1, objTabelle.getRows().Count -2). _
      setValue(dblSchichtzulage)



   objTabelle.getCellByPosition(1, objTabelle.getRows().Count -2). _
      NumberFormat = 106
End If

If dblErschwernisZulage <> 0 Then
   objTabelle.getRows().insertByIndex(objTabelle.getRows().Count -1, 1)
   objTabelle.getCellByPosition(0, objTabelle.getRows().Count -2). _
      String = "Erschwerniszulage"
   objTabelle.getCellByPosition(1, objTabelle.getRows().Count -2). _
      setValue(dblErschwernisZulage)
   objTabelle.getCellByPosition(1, objTabelle.getRows().Count -2). _
      NumberFormat = 106
End If
```

```
If dblSchweisserzulage <> 0 Then
   objTabelle.getRows().insertByIndex(objTabelle.getRows().Count -1, 1)
   objTabelle.getCellByPosition(0, objTabelle.getRows().Count -2). _
      String = "Schweißerzulage"
   objTabelle.getCellByPosition(1, objTabelle.getRows().Count -2). _
      setValue(dblSchweisserzulage)
   objTabelle.getCellByPosition(1, objTabelle.getRows().Count -2). _
      NumberFormat = 106
End If
```

Die Zahlen werden als „Value“ und nicht als „String“ in die rechte Spalte eingetragen, was an der Rechtsbündigkeit erkennbar ist. Sie werden als Währung formatiert.

Die Liste der Zahlenformate kann der Tabelle 5.22 aus Kapitel 5 entnommen werden. Allerdings scheint hier ein Bug vorzuliegen.

Wurde keine zusätzliche Information eingegeben, dann wird die letzte Spalte gelöscht:

```
   If objTabelle.getRows().Count = 2 Then
      objTabelle.getRows().removeByIndex(1, 1)
```

Man hätte übrigens auch einen Zähler mitlaufen lassen können.

Falls Werte eingegeben wurden, dann wird die Summe gezogen. Sie muss in der Form

```
=sum(<B1>|<B2>|<B3>|<B4>)
```

als Formel (`setFormula`) in die Zelle geschrieben werden:

```
   Else
      strFormel = "=sum("
      For intZeilen = 1 To objTabelle.getRows().Count - 1
         strFormel = strFormel & "<B" & intZeilen & ">|"
      Next
      strFormel = Left(strFormel, Len(strFormel) - 1) & ")"
      objTabelle.getCellByPosition(0, objTabelle.getRows().Count -1). _
         String = "Summe"
      objZellCursor = objTabelle.createCursorByCellName("A" & _
         objTabelle.getRows().Count)
      objZellCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
```

```
    objTabelle.getCellByPosition(1, objTabelle.getRows().Count -1). _
       setFormula(strFormel)
 End If
```

Der Begriff „Summe“ wird noch fett formatiert.

Übertarifliche·Zulage¶	500
Schichtzulage¶	200
Erschwerniszulage¶	50
Schweißerzulage¶	50
Summe¶	800

¶

Abbildung 6.135 Die neue Tabelle

Ein komplexeres Beispiel finden Sie in Kapitel 10.

6.13.4 Das SpaltenpProblem

Der Zugriff auf Spalten erfolgt analog dem Zeilenzugriff über die Methoden `getByIndex`, `getCount`, `insertByIndex` und `removeByIndex` am Column-Objekt, das über `getColumns` erreicht wird. Sie können jedoch nur in Tabellen angewendet werden, die keine zusammengefassten Tabellenzellen enthalten. Eine spaltenweise Formatierung von Zellen über OOoBasic ist nicht möglich. Noch ärgerlicher: Es scheint keine Möglichkeit zu existieren, die Spaltenbreite festzulegen!

6.14 Textmarken

Neben Tabellen stellen sicherlich Textmarken eine zentrale Rolle dar. Sie werden benötigt, um Textstellen eindeutig zu identifizieren und beispielsweise anzuspringen, um dort Text einzutragen. Die Sammlung der Textmarken eines Dokumentes erhält man über das Objekt `Bookmarks` oder `getBookmarks()`. Die Anzahl kann mit der Eigenschaft `Count` oder der Methode `getCount()` ermittelt werden (man kann ebenso mit `hasElements()` überprüfen, ob überhaupt Textmarken vorhanden sind), auf eine Textmarke kann über `getByName` oder `getByIndex` zugegriffen werden. Die gesamte Sammlung könnte über `getElementNames` ausgelesen werden, mit `hasByName` wird überprüft, ob eine bestimmte Textmarke existiert. Das folgende Programm (*Beispiele09.sxw*) ermittelt sämtliche Textmarken:

```
Dim obTextmarken As Object
Dim objTextmarke As Object
Dim i As Integer
Dim strTextmarken As String
obTextmarken = ThisComponent.getBookmarks()

For i = 0 To obTextmarken.Count - 1
   objTextmarke = obTextmarken.getByIndex(i)
   strTextmarken = strTextmarken & Chr(13) & objTextmarke.Name
Next
MsgBox strTextmarken
```

Um nun an einer Stelle etwas hineinzuschreiben, wird der Cursor an den Anker des Textmarken-objets gesetzt (*Beispiele09.sxw*). Dort kann nun Text geschrieben werden:

```
Dim objDatei As Object
Dim objTextmarken As Object
Dim objTextmarke As Object
Dim objCursor As Object

objDatei = ThisComponent
objTextmarken = objDatei.getBookmarks()
If objTextmarken.hasByName("Sprung") = True Then
   objTextmarke = objTextmarken.getByName("Sprung")
   objCursor = _
      objDatei.Text.createTextCursorByRange(objTextmarke.Anchor)
   objCursor.String = "Hier steh ich nun, ich armer Tor."
End If
```

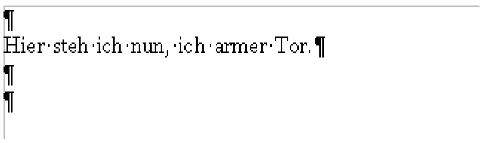


Abbildung 6.136 An einer vorhandenen Textmarke wird Text eingetragen.

Interessant erscheint folgendes Beispiel:

```
Sub Textmarke03
   Dim objDatei As Object
   Dim objTextmarken As Object
   Dim objTextmarke As Object
   Dim objCursor As Object

   objDatei = ThisComponent
   objCursor = objDatei.Text.createTextCursor()
   objCursor.gotoEnd(False)

   objTextmarken = objDatei.getBookmarks()
   If objTextmarken.hasByName("Sprung") = True Then
      objTextmarke = _
         objDatei.createInstance("com.sun.star.text.Bookmark")
      objTextmarke.setName("Sprung")
      objDatei.Text.insertTextContent(objCursor, objTextmarke, False)
   End If

End Sub
```

Es wird darin geprüft, ob eine Textmarke mit dem Namen „Sprung“ existiert. Falls ja, dann wird eine Textmarke eingefügt, die den gleichen Namen trägt. Entweder wurde vergessen, die alte Textmarke zu löschen, oder der neuen wurde fälschlicherweise der gleiche Namen gegeben. Es wäre zu vermuten, dass nun die alte Textmarke an die neue Position verschoben wird oder dass eine Fehlermeldung erscheint – jedoch passiert nichts dergleichen. Es wird eine neue Textmarke an die neue Position eingefügt – jedoch unter dem Namen „Sprung1“. Sie erhält einen anderen Namen. Lässt man das Makro einige Hundert Male laufen, dann erhält man ebenso viele unterschiedliche Textmarken, die sich nur durch die Zahl am Ende unterscheiden:

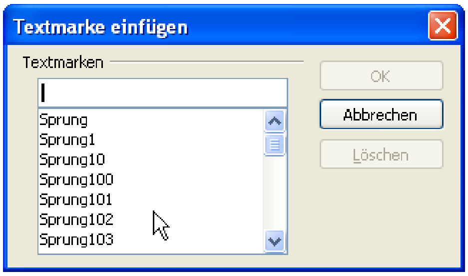


Abbildung 6.137 Neue Textmarken werden generiert.

Übrigens: Beim Löschen von 999 Textmarken ist mir OOo abgestürzt. Vielleicht sollte man solche Spielereien doch besser bleiben lassen …

Ein Beispiel für Texte, die an Textmarken geschrieben werden, finden Sie in Kapitel 10.

6.15 Textrahmen

Manchmal werden in Dokumente nicht nur Texte, Tabellen, Grafiken und andere Elemente innerhalb des Fließtextes eingefügt, sondern Elemente müssen nebeneinander angeordnet werden. Neben Tabellen stellt das Textrahmen-Objekt ein geeignetes Werkzeug hierfür zur Verfügung. Es kann programmiert werden:

```
Dim objDoc As Object
Dim objCursor As Object
Dim objTextrahmen As Object

objDoc = ThisComponent
objCursor = objDoc.Text.createTextCursor()
objTextrahmen = objDoc.createInstance("com.sun.star.text.TextFrame")
objDoc.Text.insertTextContent(objCursor, objTextrahmen, False)
```

Die Erzeugung erfolgt über die Methode `createInstance` des Dokument-Objektes. Der so erzeugte Textrahmen kann anschließend über die Methode `insertTextContent` des Textobjektes in das Dokument eingefügt werden. Dabei müssen Textcursor, Textrahmen-Objekt und die „Ausdehnung" angegeben werden. Folgende Eigenschaften und Methoden sind vom Tabellenobjekt bekannt:

BackColor, BackColorRGB, BackColorTransparency, BackGraphicFilter, BackGraphicLocation, BackGraphicTransparency, BackGraphicURL, BackTransparent, BorderDistance, BottomBorder, BottomBorderDistance, BottomMargin, EndRedline, Height, HoriOrient, HoriOrientPosition, HoriOrientRelation, HyperLinkName, HyperLinkTarget, HyperLinkURL, ImageMap, ServerMap, LeftBorder, LeftBorderDistance, LeftMargin, LinkDisplayName, PositionProtected, Print, RightBorder, RightBorderDistance, RightMargin, ShadowFormat, Size, SizeProtected, SizeType, StartRedline, TextWrap, TopBorder, TopBorderDistance, TopMargin, UserDefinedAttributes, VertOrient, VertOrientPosition, VertOrientRelation, Width, WritingMode, ZOrder, Anchor, Text, Position, Name, Start, End, String,

Die Methoden sind weitgehend bekannt:

Object getAnchor(void), Object getText(void), Object getElementType(void), Boolean hasElements(void), Object createEnumeration(void), Object getEvents(void), Array getTypes(void), Array getImplementationId(void), Object queryAdapter(void), String getImplementationName(void), Boolean supportsService(String), Array getSupportedServiceNames(void), Object getPropertySetInfo(void), Void setPropertyValue(String, Variant), Empty getPropertyValue(String), Void addPropertyChangeListener(String, Object), Void removePropertyChangeListener(String, Object), Void addVetoableChangeListener(String, Object), Void removeVetoableChangeListener(String, Object), Long getPropertyState(String), Array getPropertyStates(Array), Void setPropertyToDefault(String), Empty getPropertyDefault(String), String getShapeType(void), Object getPosition(void), Void setPosition(Object), Object getSize(void), Void setSize(Object), String getName(void), Void setName(String), Unknown , -Type! getSomething(Array), Object getText(void), Object getStart(void), Object getEnd(void), String getString(void), Void setString(String), Object createTextCursor(void), Object createTextCursorByRange(Object), Void insertString(Object, String, Boolean), Void insertControlCharacter(Object, Integer, Boolean), Void insertTextContent(Object, Object, Boolean), Void removeTextContent(Object), Object getText(void), Object getText(void), Integer compareRegionStarts(Object, Object), Integer compareRegionEnds(Object, Object), Void insertTextContentBefore(Object, Object), Void insertTextContentAfter(Object, Object), Void removeTextContentBefore(Object), Void removeTextContentAfter(Object)

Tabelle 6.112 Eigenschaften, die nur dem Textrahmenobjekt vorliegen

Eigenschaft	**Wert**	**Beschreibung**
AnchorFrame	Object/void	der Ankerrahmen
AnchorPageNo	Integer	die Seitennummer des Textankers
AnchorType	Long	eine Konstante des Typs com.sun.star.text.TextContentAnchorType: AS_CHARACTER (1), AT_CHARACTER (4), AT_PARA-

Eigenschaft	Wert	Beschreibung
		GRAPH (0), AT_FRAME(3) und AT_PAGE(2)
AnchorTypes	Array	Sammlung der Textanker
ChainNextName und ChainPrevName	String/void	bei verketteten Textrahmen der nächsten, bzw. vorhergehende Rahmen
ContentProtected	Boolean	geschützter Inhalt
FrameHeightAbsolute	Long	absolute Höhe
FrameHeightPercent	Integer	prozentuale Höhe
FrameIsAutomaticHeight	Boolean	automatische Höhe
FrameStyleName	String	Name des Stils
FrameWidthAbsolute	Long	absolute Breite
FrameWidthPercent	Integer	prozentuale Breite
IsSyncHeightToWidth	Boolean	die Rahmenhöhe folgt der Breite
IsSyncWidthToHeight	Boolean	die Rahmenbreite folgt der Höhe
RelativeHeight	Integer	die relative Höhe
RelativeWidth	Integer	die relative Breite
Surround	Long	Typ des umfließenden Textes. Eine Aufzählung des Typs: com.sun.star.Text.WrapTextMode: NONE, THROUGHT, PARALLEL, DYNAMIC, LEFT and RIGHT
SurroundAnchorOnly	Boolean	legt fest, ob der Text des Absatzes, in dem sich der Anker befindet, um den Textrahmen fließt
TextColumns	Object	Spalten
PageToggle		das Objekt wird auf geraden und ungeraden Seiten gespiegelt
Opaque		durchsichtig oder ausgefüllt

Das folgende Beispiel (*Beispiele10.sxw*) erstellt einen Textrahmen, in dem der Textcursor vier Absätze nach unten wandert:

```
Sub Textrahmen2

   Dim objDoc As Object
   Dim TextTables As Object
   Dim objCursor As Object
   Dim objTextrahmen As Object
```

```
Dim i As Integer

objDoc = ThisComponent
objCursor = objDoc.Text.createTextCursor()
For i = 1 To 4
   objCursor.gotoNextParagraph(False)
Next

objTextrahmen = objDoc.createInstance("com.sun.star.text.TextFrame")
```

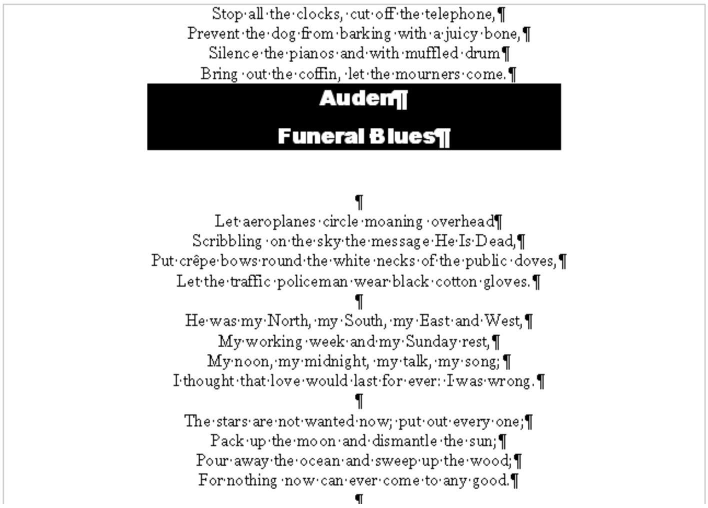
Der Textrahmen erhält eine feste Breite und Höhe, einen Anfangs- und Endabstand, wird an den Absatz gebunden, auf der Seite zentriert und schwarz formatiert. Der Text des Absatzes fließt nicht um das Rahmenobjekt herum:

```
With objTextrahmen
   .Width = 10000
   .Height = 1600
   .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PARAGRAPH
   .TopMargin = 1000
   .BottomMargin = 1000
   .LeftMargin = 0
   .RightMargin = 0
   .BorderDistance = 0
   .BackColor = RGB(0, 0, 0)
   .HoriOrient = com.sun.star.text.HoriOrientation.CENTER
   .VertOrient = com.sun.star.text.VertOrientation.LINE_TOP
   .Surround = com.sun.star.Text.WrapTextMode.NONE
End With
objDoc.Text.insertTextContent(objCursor, objTextrahmen, False)
```

Nun kann in dem Textrahmen ein Textcursor erstellt werden, mit dessen Hilfe Text geschrieben und formatiert wird:

```
objCursor = objTextrahmen.createTextCursor()
With objCursor
   .ParaAdjust = com.sun.star.style.ParagraphAdjust.CENTER
   .CharColor = RGB(255, 255, 255)
```

```
        .CharFontName = "Arial Black"
        .CharHeight = 14
        .CharWeight = com.sun.star.awt.FontWeight.BOLD

        .String = "Auden"
        .gotoEnd(False)
        objDoc.Text.insertControlCharacter(objCursor, 0, False)
        .String = "Funeral Blues"
    End With
End Sub
```


Stop all the clocks, cut off the telephone,¶
Prevent the dog from barking with a juicy bone,¶
Silence the pianos and with muffled drum¶
Bring out the coffin, let the mourners come.¶
Auden¶
Funeral Blues¶

¶
Let aeroplanes circle moaning overhead¶
Scribbling on the sky the message He Is Dead,¶
Put crêpe bows round the white necks of the public doves,¶
Let the traffic policeman wear black cotton gloves.¶
¶
He was my North, my South, my East and West,¶
My working week and my Sunday rest,¶
My noon, my midnight, my talk, my song;¶
I thought that love would last for ever: I was wrong.¶
¶
The stars are not wanted now; put out every one;¶
Pack up the moon and dismantle the sun;¶
Pour away the ocean and sweep up the wood;¶
For nothing now can ever come to any good.¶
¶

Abbildung 6.138 Ein Textrahmen mit Text wird eingefügt und formatiert.

Der Text kann erst in den Textrahmen geschrieben werden, nachdem er mit `insert-TextContent` erstellt wurde. Auch die Eigenschaft `String` und die Methode `set-String` versagen, wenn Sie versuchen, Text in das noch nicht im Dokument erstellte Objekt einzufügen.

So kann man mit Hilfe eines Textrahmens Bilder hinter einen Text legen (*Beispiele11.sxw*):

```
Sub Textrahmen3
```

```
Dim objDoc As Object
Dim strPfad() As String
Dim strBild As String
Dim i As Integer
Dim objCursor As Object
Dim objTextrahmen As Object
Dim objRandLinie As New com.sun.star.table.BorderLine

objDoc = ThisComponent
objCursor = objDoc.Text.createTextCursor()
```

Der Speicherort der Datei wird ermittelt, denn dort befindet sich das Bild, das eingefügt wird:

```
strPfad = Split(objdoc.URL, "/")
strPfad(UBound(strPfad)) = ""
strBild = Join(strPfad, "/") & "PIC00017.jpg"
```

Die Linienstärke wird ausgeschaltet:

```
objRandLinie.InnerLineWidth = 0
objRandLinie.OuterLineWidth = 0
```

Das Rahmenobjekt wird erzeugt:

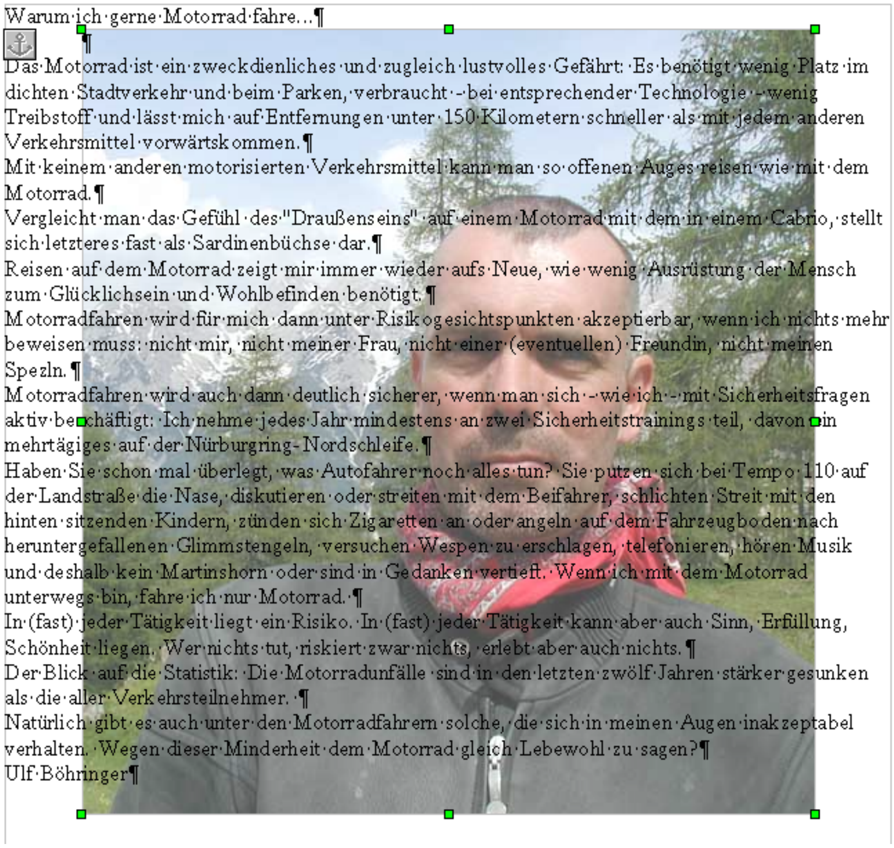
```
objTextrahmen = objDoc.createInstance("com.sun.star.text.TextFrame")

With objTextrahmen
   .Width = 10000
   .Height = 16000
   .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PARAGRAPH
   .TopMargin = 0
   .BottomMargin = 0
   .LeftMargin = 0
   .RightMargin = 0
   .BorderDistance = 0

   .HoriOrient = com.sun.star.text.HoriOrientation.CENTER
```

```
      .VertOrient = com.sun.star.text.VertOrientation.LINE_TOP
      .Surround = com.sun.star.Text.WrapTextMode.THROUGHT
      .BackGraphicTransparency = 30

      .LeftBorder = objRandLinie
      .RightBorder = objRandLinie
      .TopBorder = objRandLinie
      .BottomBorder = objRandLinie
```

Falls das Bild in dem entsprechenden Ordner existiert, dann wird es eingefügt:

```
      If Dir(strBild, 0) <> "" Then
         .BackGraphicURL = strBild
      End If
      .Opaque = False
   End With

   objDoc.Text.insertTextContent(objCursor, objTextrahmen, False)
End Sub
```

Beachten Sie, dass die Eigenschaft „Opaque“ auf False gesetzt wird, was im Dialog der Eigenschaft „im Hintergrund“ im Registerblatt „Umlauf“ entspricht.



Warum ich gerne Motorrad fahre...

Das Motorrad ist ein zweckdienliches und zugleich lustvolles Gefährt: Es benötigt wenig Platz im dichten Stadtverkehr und beim Parken, verbraucht - bei entsprechender Technologie - wenig Treibstoff und lässt mich auf Entfernungen unter 150 Kilometern schneller als mit jedem anderen Verkehrsmittel vorwärtskommen.

Mit keinem anderen motorisierten Verkehrsmittel kann man so offenen Auges reisen wie mit dem Motorrad.

Vergleicht man das Gefühl des "Draußenseins" auf einem Motorrad mit dem in einem Cabrio, stellt sich letzteres fast als Sardinenbüchse dar.

Reisen auf dem Motorrad zeigt mir immer wieder aufs Neue, wie wenig Ausrüstung der Mensch zum Glücklichsein und Wohlbefinden benötigt.

Motorradfahren wird für mich dann unter Risikogesichtspunkten akzeptierbar, wenn ich nichts mehr beweisen muss: nicht mir, nicht meiner Frau, nicht einer (eventuellen) Freundin, nicht meinen Spezln.

Motorradfahren wird auch dann deutlich sicherer, wenn man sich - wie ich - mit Sicherheitsfragen aktiv beschäftigt: Ich nehme jedes Jahr mindestens an zwei Sicherheitstrainings teil, davon ein mehrtägiges auf der Nürburgring-Nordschleife.

Haben Sie schon mal überlegt, was Autofahrer noch alles tun? Sie putzen sich bei Tempo 110 auf der Landstraße die Nase, diskutieren oder streiten mit dem Beifahrer, schlichten Streit mit den hinten sitzenden Kindern, zünden sich Zigaretten an oder angeln auf dem Fahrzeugboden nach heruntergefallenen Glimmstengeln, versuchen Wespen zu erschlagen, telefonieren, hören Musik und deshalb kein Martinshorn oder sind in Gedanken vertieft. Wenn ich mit dem Motorrad unterwegs bin, fahre ich nur Motorrad.

In (fast) jeder Tätigkeit liegt ein Risiko. In (fast) jeder Tätigkeit kann aber auch Sinn, Erfüllung, Schönheit liegen. Wer nichts tut, riskiert zwar nichts, erlebt aber auch nichts.

Der Blick auf die Statistik: Die Motorradunfälle sind in den letzten zwölf Jahren stärker gesunken als die aller Verkehrsteilnehmer.

Natürlich gibt es auch unter den Motorradfahrern solche, die sich in meinen Augen inakzeptabel verhalten. Wegen dieser Minderheit dem Motorrad gleich Lebewohl zu sagen?

Ulf Böhringer

Abbildung 6.139 Ein Textrahmen mit Bild

6.16 Textfelder

Textfelder zählen zu den TextContent-Objekten, da sie über den reinen Text hinaus eine zusätzliche Logik zur Verfügung stellen. Textfelder lassen sich über die gleichen Methoden in ein Textdokument einfügen wie andere Objekte, wie beispielsweise Tabellen, Textrahmen, Grafiken oder Abschnitte:

```
Dim objDoc As Object
Dim objDatFeld As Object
Dim objCursor As Object
```

```
objDoc = ThisComponent
objCursor = objDoc.Text.createTextCursor()

objDatFeld = _
   objDoc.createInstance("com.sun.star.text.TextField.DateTime")
objDatFeld.IsFixed = False
objDatFeld.IsDate = True
objDoc.Text.insertTextContent(objCursor, objDatFeld, False)
```

Das Beispiel fügt am Anfang des aktuellen Textdokumentes ein Textfeld mit dem aktuellen Datum ein. Wird die Eigenschaft IsDate auf „True“ gesetzt, dann wird das Datum und nicht die Urhzeit angezeigt. Die Eigenschaft IsFixed = False aktualisiert das Datum, wenn das Dokument erneut geöffnet wird.

Tabelle 6.113 OOo stellt folgende Textfelder des Services „com.sun.star.text.TextField“ zur Verfügung:

Textfeld	**Beschreibung**	**Wichtige Eigenschaften**	**Wert**
Annotation	Notizen (wird über Einfügen | Notiz) eingefügt.	Author Content	String String
Author	der Autor	Content CurrentPresentation FullName IsFixed	String String Boolean Boolean
Bibliography	Verzeichnisse (wird über Einfügen | Verzeichnisse) eingefügt	Fields	Array
Chapter	Kapitel	ChapterFormat (Konstante gemäß com.sun.star.text.ChapterFormat: NAME, NUMBER, NAME_NUMBER, NO_PREFIX_SUFFIX und DIGIT) Level	Integer Integer
CharacterCount	Anzahl der Zeichen	NumberingType	Integer
CombinedCharacters	Zeichen kombinieren (verbindet bis zu sechs Zeichen zu einem Zeichen auf	Content	String

Textfeld	Beschreibung	Wichtige Eigenschaften	Wert
	zwei Zeilen)		
ConditionalText	versteckter Text	Condition FalseContent IsConditionTrue TrueContent	String String Boolean String
Database	Datenbank	Content CurrentPresentation DataBaseFormat IsVisible NumberFormat	String String Boolean Boolean Long
DatabaseName	Name der Datenbank	DataBaseName DataCommandType (Konstante gemäß com.sun.star.sdb.CommandType: TABLE, QUERY und COMMAND) DataTableName IsVisible	String Long String Boolean
DatabaseNextSet	Auswahl	Condition DataBaseName DataCommandType DataTableName	String String Long String
DatabaseNumberOfSet	Auswahl	Condition DataBaseName DataCommandType DataTableName SetNumber	String String Long String Long
DatabaseSetNumber	Auswahl	DataBaseName DataCommandType DataTableName IsVisible NumberingType SetNumber	String Long String Boolean Integer Long
DateTime	Datum/Uhrzeit	IsDate IsFixed IsFixedLanguage	Boolean Boolean Boolean

Textfeld	Beschreibung	Wichtige Eigenschaften	Wert
		NumberFormat	Long
DDE	DDE-Verbindung		
docinfo verfügt über folgende Objekte:	scheint nicht zu gehen		
ChangeAuthor			
ChangeDateTime			
CreateAuthor			
.CreateDateTime			
Description			
EditTime			
Info0, Info1, Info2 und Info3			
Keywords			
PrintAuthor			
PrintDateTime			
Revision			
Subject			
Title			
EmbeddedObjectCount	die Anzahl der eingebetteten Objekte	NumberingType	Integer
ExtendedUser	der Benutzer, der unter Extras | Optionen | OOo oder StarOffice | Benutzerdaten eingetragen ist	Content CurrentPresentation IsFixed UserDataTyp (Konstante gemäß com.sun.star.text.UserDataPart: COMPANY, FIRSTNAME, NAME, SHORTCUT, STREET, COUNTRY, ZIP, CITY, TITLE, POSITION, PHONE_PRIVATE, PHONE_COMPANY, FAX, EMAIL und STATE)	String String Boolean Integer
FileName	Dateiname (URL)	CurrentPresentation FileFormat IsFixed	String Integer Boolean
GetExpression	Variable	Content CurrentPresentation	String String

Textfeld	Beschreibung	Wichtige Eigenschaften	Wert
		IsFixedLanguage IsShowFormula NumberFormat SubType TextWrap Value VariableSubtype	Boolean Boolean Long Integer Long Double Integer
GetReference	wird über Einfügen | Querverweis eingefügt	CurrentPresentation ReferenceFieldPart (Konstante gemäß com.sun.star.text ReferenceFieldPart: PAGE, CHAPTER, TEXT, UP_DOWN, PAGE_DESC, CATEGORY_AND_NUMBER, ONLY_CAPTION und ONLY_SEQUENCE_NUMBER) ReferenceFieldSource (Konstante gemäß com.sun.star.text .ReferenceFieldSource: REFERENCE_MARK, SEQUENCE_FIELD, BOOKMARK, FOOTNOTE und ENDNOTE) SequenceNumber SourceName	String Integer Integer Integer String
GraphicObjectCount	Anzahl der Grafik-Objekte	NumberingType	Integer
HiddenParagraph	verborgene Absätze	Condition IsHidden	String Boolean
HiddenText	verborgener Text	Condition Content IsHidden	String String Boolean
Input	Eingabefeld	Content Hint	String String
InputUser	Eingabeliste	Content	String

Textfeld	Beschreibung	Wichtige Eigenschaften	Wert
		Hint	String
JumpEdit	Platzhalter	Hint PlaceHolder PlaceHolderType (Konstante gemäß com.sun.star.text.PlaceHolderType: TEXT, TABLE, TEXTFRAME, GRAPHIC und OBJECT)	String String Integer
Macro	Makro	Hint MacroLibrary MacroName	String String String
PageCount	Anzahl der Seiten	NumberingType	Integer
PageNumber	Seitennummer	NumberingType Offset SubType (Konstante gemäß com.sun.star.text.PageNumberType: PREV, CURRENT und NEXT) TextWrap UserText	Integer Integer Long Long String
ParagraphCount	Anzahl der Absätze	NumberingType	Integer
ReferencePageGet	Seitennummer einer Referenz	CurrentPresentation NumberingType	String Integer
ReferencePageSet	Seitennummer einer Referenz	Offset On	Integer Boolean
Script	Skript	Content ScriptType URLContent	String String Boolean
SetExpression	Formel einfügen	Content CurrentPresentation Hint Input IsFixedLanguage IsShowFormula IsVisible NumberFormat NumberingType	String String String Boolean Boolean Boolean Long Integer Integer

Textfeld	Beschreibung	Wichtige Eigenschaften	Wert
		SequenceValue SubType TextWrap Value VariableName	Integer Long Double String
TableCount	Anzahl der Tabellen	NumberingType	Integer
TableFormula	Formel innerhalb einer Tabelle	CurrentPresentation IsShowFormula NumberFormat	String Boolean Long
TemplateName	Name der Vorlage	FileFormat	Integer
User	Benutzerdefiniertes Feld	IsFixedLanguage IsShowFormula IsVisible NumberFormat	Boolean Boolean Boolean Long
WordCount	Anzahl der Wörter	NumberingType	Integer

Tabelle 6.114 Die Konstanten des Typs com.sun.star.style.NumeringType lauten:

Konstante	Bedeutung
CHARS_UPPER_LETTER	Großbuchstaben: „A, B, C, D ..."
CHARS_LOWER_LETTER	Kleinbuchstaben: „a, b, c, e..."
ROMAN_UPPER	römische Zahlen; „I, II, III, IV ..."
ROMAN_LOWER	kleine römische Zahlen: „i, ii, iii, iv ..."
ARABIC	arabische Zahlen: „1, 2, 3, 4 ..."
NUMBER_NONE	Nummerierung ist ausgeschaltet
CHAR_SPECIAL	spezieller Zeichensatz
PAGE_DESCRIPTOR	wird im Seitenstil festgelegt
BITMAP	wird als Bitmap dargestellt
CHARS_UPPER_LETTER_N	Großbuchstaben: „A, B ... Y, Z, AA, BB, CC ... AAA ..."
CHARS_LOWER_LETTER_N	Kleinbuchstaben: „a, b ... y, z, aa, bb, cc ... aaa ..."
TRANSLITERATION	Ein Transliterationsmodul wird für Zahlen des Chinesischen, Japanischen, … benötigt
NATIVE_NUMBERING	Der „nativenumbersupplier service" wird aufgerufen
FULLWIDTH_ARABIC	„arabische" Zahlen aus arabischen Ländern

Konstante	Bedeutung
CIRCLE_NUMBER	Nummerierung mit Kreisen um die Zahlen
NUMBER_LOWER_ZH	Nummerierung für chinesische Kleinbuchstaben
NUMBER_UPPER_ZH	Nummerierung für chinesische Großbuchstaben
NUMBER_UPPER_ZH_TW	Nummerierung für traditionelle chinesische Kleinbuchstaben
TIAN_GAN_ZH	Nummerierung für chinesische Tian Gan
DI_ZI_ZH	Nummerierung für chinesische Di Zi
NUMBER_TRADITIONAL_JA	Nummerierung für japanische traditionelle Zahlen
AIU_FULLWIDTH_JA	Zeichen für japanische AIU-Vollzeichen
AIU_HALFWIDTH_JA	Zeichen für japanische AIU-Halbzeichen
IROHA_FULLWIDTH_JA	Zeichen für japanische IROHA-Vollzeichen
IROHA_HALFWIDTH_JA	Zeichen für japanische IROHA-Halbzeichen
NUMBER_UPPER_KO	Zeichen für koreanische Großbuchstaben
NUMBER_HANGUL_KO	Zeichen für koreanische Hangul-Zahlen
HANGUL_JAMO_KO	Zeichen für koreanische Hangul-Jamo
HANGUL_SYLLABLE_KO	Zeichen für koreanische Hangul-Syllable
HANGUL_CIRCLED_JAMO_KO	Zeichen für koreanische Großbuchstaben Hangul-Kreis-Jamo
HANGUL_CIRCLED_SYLLABLE_KO	Zeichen für koreanische Hangul-Kreis-Silben
CHARS_ARABIC	arabische Buchstaben
CHARS_THAI	Thai-Buchstaben

Das folgende Beispiel (*Beispiele13.sxw*) zeigt, wie man eine Seitenzahl in die Fußzeile eines Dokumentes einfügt.

```
Sub Fusszeile
   Dim objDoc As Object
   Dim objStil As Object
   Dim objFuss As Object
   Dim objCursor As Object
   Dim objFeld As Object

   objDoc = ThisComponent
   objFeld = _
      objDoc.createInstance("com.sun.star.text.TextField.PageNumber")
```

```
    objFeld.NumberingType = com.sun.star.style.NumberingType.ARABIC
    objFeld.SubType = com.sun.star.text.PageNumberType.CURRENT

    objStil = objDoc.StyleFamilies.getByName("PageStyles")
    objFuss = objStil("Standard")
    objFuss.FooterIsOn = True

    objCursor = objFuss.FooterTextLeft.Text.createTextCursor()
    objCursor.String = "Seite "
    objCursor.gotoEndOfParagraph(False)
    objFuss.FooterTextLeft.Text.insertTextContent _
       (objCursor, objFeld, False)

    objCursor.String = " von "
    objCursor.gotoEndOfParagraph(False)

    objFeld = _
       objDoc.createInstance("com.sun.star.text.TextField.PageCount")
    objFeld.NumberingType = com.sun.star.style.NumberingType.ARABIC
    objFuss.FooterTextLeft.Text.insertTextContent _
       (objCursor, objFeld, False)
End Sub
```

Das Beispiel erzeugt zunächst ein Textfeld, das den Service „com.sun.star.text.TextField.PageNumber“ unterstützt. Da die Kopf- und Fußzeilen als Teil der Seitenvorlagen von OOo definiert sind, wird dies zunächst über die Liste aller PageStyles ermittelt.

Um sicherzugehen, dass die Fußzeile tatsächlich sichtbar ist, wird die Eigenschaft `FooterIsOn` auf „True“ gesetzt. Anschließend wird das Textfeld über das zugehörige Text-Objekt der linken Fußzeile in das Dokument eingefügt.

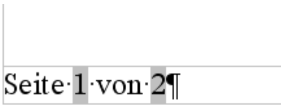


Abbildung 6.140 Eine neue Fußzeile

Analog könnte der Dateiname in die Fußzeile geschrieben werden:

```
   objFeld = _
      objDoc.createInstance("com.sun.star.text.TextField.FileName")
   objFeld.FileFormat = com.sun.star.text.FilenameDisplayFormat.FULL
   objCursor = objFuss.FooterTextLeft.Text.createTextCursor()
   objFuss.FooterTextLeft.Text.insertTextContent(objCursor, _
      objFeld, False)
```

Da Notizen auch Felder sind, können sie ebenso eingefügt werden (*Beispiele13.sxw*):

```
Sub Notiz_Einfuegen
   Dim objDoc As Object
   Dim objMarkierungen As Object
   Dim objCursor As Object
   Dim objNotiz As Object

   objDoc = ThisComponent
   objMarkierungen = objDoc.getCurrentSelection()

   If objMarkierungen.Count > 1 Then
      Msgbox "Bitte keine Mehrfachmarkierung!"
   Else
      objMarkierung = objMarkierungen(0)
      objCursor = objDoc.Text.createTextCursorByRange(objMarkierung)
      If Not objCursor.IsCollapsed() Then
         MsgBox "Bitte nichts markieren!"
      Else
         objNotiz = _
```

```
        objDoc.createInstance("com.sun.star.text.TextField.Annotation")
         objNotiz.Content = "Hier muss noch etwas eingefügt werden!"
         objMarkierungen(0).Text.insertTextContent(objCursor, _
            objNotiz, False)
      End If
   End If

End Sub
```

Da in diesem Beispiel an die Stelle des Textcursors eine Notiz eingefügt wird, wird zuerst überprüft, ob nicht mehrere Stellen markiert sind und ob der Cursor nicht ausgedehnt ist. Danach wird ein Notizfeld mit einem Textinhalt erstellt und an die Poition der Markierung eingefügt.

Mit diesem Wissen können leicht sämtliche Notizen aus einem Dokument ausgelesen und beispielsweise in ein anderes Dokument geschrieben werden:

```
Sub Notizen_Einsammeln
   Dim objDoc As Object
   Dim objNotizen As Object
   Dim objNotiz As Object
   Dim strNotiz As String

   objDoc = ThisComponent
   objNotizen = objDoc.getTextFields().createEnumeration()
   intWeiter = 99
   Do While (objNotizen.hasMoreElements() And intWeiter <> 7)
      objNotiz = objNotizen.nextElement()
      If objNotiz.supportsService _
         ("com.sun.star.text.TextField.Annotation") Then
         strNotiz = strNotiz & Chr(13) & objNotiz.Content
      End If
   Loop

   MsgBox strNotiz
End Sub
```

Jedoch scheint es nicht möglich zu sein, die Seitennummerierung der entsprechenden Seiten auszulesen.

Dafür ist es jedoch möglich, sämtliche Notizen innerhalb eines Dokumentes nacheinander anzuspringen (*Beispiele13.sxw*):

```
Sub Notizen_Anspringen
   Dim objDoc As Object
   Dim objNotizen As Object
   Dim objNotiz As Object
   Dim intWeiter As Integer

   objDoc = ThisComponent
   objNotizen = objDoc.getTextFields().createEnumeration()
   intWeiter = 99
   Do While (objNotizen.hasMoreElements() And intWeiter <> 7)
      objNotiz = objNotizen.nextElement()
      If objNotiz.supportsService _
         ("com.sun.star.text.TextField.Annotation") Then
         ThisComponent.CurrentController.getViewCursor().gotoRange _
            (objNotiz.Anchor, False)
         intWeiter = MsgBox("Weitersuchen?", 4)
      End If
   Loop
   If intWeiter <> 7 Then
      MsgBox "Keine Notiz mehr gefunden."
   End If
End Sub
```

Zugriff auf Writer

Vielleicht nimmt dieses Programm – oder besser: diese beiden Programme – den geringsten Stellenwert bei der Programmierung ein. Dennoch sind Draw und Impress nicht zu unterschätzen, was die Möglichkeiten der Automatisierung anbelangt.

7 Draw und Impress (Präsentation und Zeichnung)

Die beiden Programme Draw und Impress unterscheiden sich nur geringfügig voneinander. In beiden Programmen geht es um Vektorgrafiken. Impress kann gegenüber Draw mehrere Seiten mit Seitenübergängen verwalten. Um herauszufinden, ob ein Dokument ein Draw- oder Impress-Dokument ist, kann mit dem SupportsService gearbeitet werden (*Beispiel01.sxi*):

```
If ThisComponent.SupportsService _
   ("com.sun.star.presentation.PresentationDocument") Then
   MsgBox "ich bin ein Impress-Dokument"
ElseIf ThisComponent.SupportsService _
   ("com.sun.star.drawing.DrawingDocument") Then
   MsgBox "ich bin ein Draw-Dokument"
Else
   MsgBox "Ich bin etwas anderes"
End If
```

Beachten Sie, dass Impress auch den Drawing-Service unterstützt – folglich müssen Sie beim Überprüfen zuerst „Presentation“ und anschließend „Drawing“ abfragen – nicht umgekehrt! Daran sieht man deutlich, dass es sich bei den Drawing-Dateien um Sonderformen von Präsentationen handelt.

7.1 Blätter

7.1.1 Seiten

Um auf die vorhandenen Seiten zuzugreifen, wird die Eigenschaft `DrawPages` oder die Methode `getDrawPages()` verwendet. Sie können mit `Count` oder `getCount()` gezählt werden, `ElementNames` und `getElementNames()` übergeben an einen Array sämtliche Namen, `hasElements()` überprüft, ob ein bestimmtes Blatt vorhanden ist, `getByName()` greift über den Namen auf ein Blatt zu, `getByIndex()` über die Indexzahl. `InsertNewByIndex()` fügt ein neues Blatt ein, und `Remove` löscht ein vorhandenes.

Tabelle 7.115 Die wichtigsten Methoden des Seitenobjekts

Methode	Wert	Beschreibung
getCount()	Long	die Anzahl der Blätter
getByIndex (Long) getByName (String)	Empty	greift auf ein Blatt zu
getElementNames (Void)	Array	greift auf alle Blätter zu
hasByName (String)	Boolean	überprüft, ob ein bestimmtes Blatt vorhanden ist
hasElements (Void)		überprüft, ob Blätter vorhanden sind (obsolet, da es keine Präsentation ohne Blätter geben kann)
remove (Object)	Void	löscht ein Blatt
insertNewByIndex (Long)	Object	fügt ein Blatt an die Position Index ein

Die folgende Prozedur fügt ein Blatt hinzu:

```
    Dim objBlatt As Object
    objBlatt = ThisComponent.getDrawPages().insertNewByIndex(0)
    objBlatt.setName("Seite1")
```

Erstaunlicherweise ist es per Programmierung möglich, ein neues Blatt wie ein bereits vorhandenes zu nennen. Dies geht in Impress nicht!

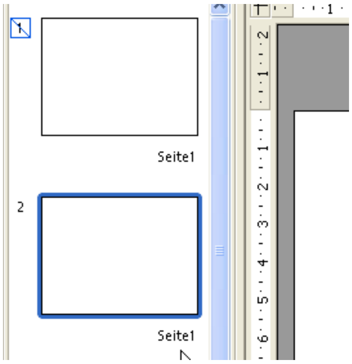


Abbildung 7.141 Per Programmierung können zwei Seiten mit dem gleichen Namen erzeugt werden.

Sinnvollerweise sollte überprüft werden, ob ein Blattname bereits existiert und ein neues Blatt anders benannt werden:

```
Sub NeueSeite2

    Dim objBlatt As Object
```

```
    Dim i As Integer
    i = 0
    objBlatt = ThisComponent.getDrawPages(). _
       insertNewByIndex(ThisComponent.getDrawPages().Count -1)
    Do
       i = i + 1
    Loop Until ThisComponent.getDrawPages().hasByName("Seite" & i) = False
    objBlatt.setName("Seite" & i)
End Sub
```

7.1.2 Hintergründe

Auch wenn ein Impress-Dokument nur einen Hintergrund haben kann, liefert die Methode `getMasterPages()` die Sammlung der (Hintergrund-)Blätter. Sie verfügen über die gleichen Methoden wie die Seiten. Und bei den Hintergründen macht die Methode `hasElements()` Sinn, da ein Impress-Dokument nicht notwendigerweise über ein Hintergrund-Blatt verfügen muss.

Und so kann einem Blatt der Hintergrund ausgeschaltet werden:

```
objBlatt.IsBackgroundObjectsVisible = False
objBlatt.IsBackgroundVisible = False
```

7.2 Ein Blatt

Tabelle 7.116 Wichtige Eigenschaften eines Blattes sind

Eigenschaft	Wert	Beschreibung
Background	Object / Void	der Hintergrund
BookmarkURL	String	
BorderBottom, BorderLeft, BorderRight und BorderTop	Long	die vier Ränder
Height und Width	Long	Höhe und Breite in 0,001 mm
Orientation	Long	die Ausrichtung: Konstante gemäß com.sun.star.view.PaperOrientation: PORTRAIT und LANDSCAPE
Visible	Boolean	sichtbar oder unsichtbar
IsBackgroundDark	Boolean	Hintergrund dunkel

Eigenschaft	Wert	Beschreibung
IsBackgroundObjectsVisible	Boolean	die Hintergrundobjekte sichtbar
IsBackgroundVisible	Boolean	der Hintergrund sichtbar
Layout	Integer	das Layout
Number	Integer	die Nummer des Zeichenblattes
Change	Long	Folienübergang (siehe Tabelle 7.18)
Duration	Long	Dauer des Folienübergangs
Effect	Long	Effekt beim Folienübergang
Sound	String	Sound
Speed	Long	Geschwindigkeit beim Folienübergang

Bei der Eigenschaft `Visible` scheint ein Fehler vorzuliegen. Zwar kann ein Blatt per Programmierung auf `Visible = False` gesetzt werden, diese Eigenschaft kann auch überprüft werden, aber es passiert dennoch nichts! Das folgende Beispiel tut nichts:

```
Sub Unsichtbar
   Dim objDoc As Object
   Dim objBlaetter As Object
   Dim objBlatt As Object

   objDoc = ThisComponent
   objBlaetter = objDoc.getDrawPages()
   objBlatt = objBlaetter.getByIndex(0)
   objBlatt.Visible = False

End Sub
```

Ebenso scheint bei `Orientation` ein Fehler vorzuliegen. Zwar überprüft sie die Ausrichtung korrekt, kann sie aber nicht setzen.

7.3 Objekte auf der Seite

Neben den bereits besprochenen Methoden einer Seite – `hasElements()`, `getCount()`, `getByIndex(Long)`, `getName()`, `setName(String)`, `getMasterPage()` und `setMasterPage(Object)` – verfügt das Seitenobjekt über die Methoden `add(Object)` und `remove(Object)`. Sie fügen ein Zeichenobjekt (Rechteck, Kreis …), Linien und Textobjekte hinzu. Diese Objekte unterstützen den Service `com.sun.star.drawing.Shape`,

Das folgende Beispiel (*Beispiel02.sxi*) erstellt einen Stern:

```
Sub Stern

   Dim objDoc As Object
   Dim objSeite As Object
   Dim objStern As Object
   Dim PolyPolygon As Variant
   Dim Punkte(10) As New com.sun.star.awt.Point


   objDoc = ThisComponent
   objSeite = objDoc.drawPages(0)
   objStern = _
      objDoc.createInstance("com.sun.star.drawing.PolyPolygonShape")
   objSeite.add(objStern)
```

Bis hierher würde man noch nichts erkennen, da das Objekt lediglich aus einem Punkt gezeichnet wurde. Deshalb werden ihm „Attribute“ zugewiesen. Zuerst die Punkte:

```
   Punkte(0).x = 4000
   Punkte(1).x = 5000
   Punkte(2).x = 8000
   Punkte(3).x = 5500
   Punkte(4).x = 6500
   Punkte(5).x = 4000
   Punkte(6).x = 1500
   Punkte(7).x = 2500
   Punkte(8).x = 0
```

```
Punkte(9).x = 3000
Punkte(10).x = 4000

Punkte(0).y = 0
Punkte(1).y = 3000
Punkte(2).y = 3000
Punkte(3).y = 4750
Punkte(4).y = 7500
Punkte(5).y = 6000
Punkte(6).y = 7500
Punkte(7).y = 4750
Punkte(8).y = 3000
Punkte(9).y = 3000
Punkte(10).y = 0
objStern.PolyPolygon = Array(Punkte())
```

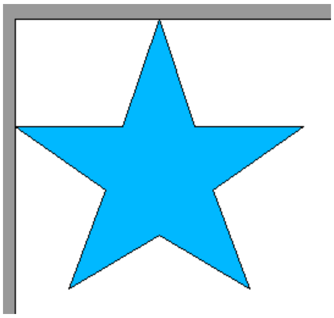


Abbildung 7.142 Der Stern in der Standardfarbe

Anschließend die Farbe Gelb:

```
objStern.FillColor = RGB(255, 255, 0)
```

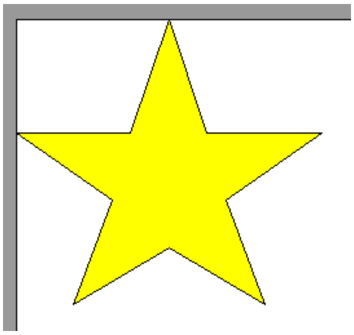


Abbildung 7.143 Der gelbe Stern

Und mit ein wenig mathematischen Kenntnissen können die zwölf Sterne des Banners der EU erzeugt werden.

7.4 Die Shapes

Betrachten wir die einzelnen Zeichnungsobjekte der Reihe nach:

Tabelle 7.117 Folgende Shapes können über com.sun.star.drawing erzeugt werden:

Shape	Beschreibung
ClosedBezierShape	geschlossene Kurve, Freihandkurve
ConnectorShape	Verbindungslinie
ControlShape	Kontrollelement
EllipseShape	Kreis, Ellipse und Bogen
GraphicObjectShape	grafisches Objekt
GroupShape	Gruppe aus mehreren Objekten
LinieShape	einfache Linie
MeasureShape	Maßlinie
OLE2Shape	ein OLE-Objekt
OpenBezierShape	offene Bézier-Kurve, Freihandkurve
PageShape	Seitenvorschau
PluginShape	eingefügtes Media-Objekt
PolyLineShape	offener Streckenzug
PolyPolygonBezierShape	Streckenzug, der aus Bézierkurven besteht
PolyPolygonShape	geschlossener Streckenzug

Shape	Beschreibung
RectangleShape	Recheck und Quadrat
TextShape	Textfeld
TitleTextShape	Textfeld, das für den Titel reserviert ist
SubtitleShape	Textfeld, das für den Untertitel reserviert ist
OutlinerShape	Textfeld für Outline-Objekte
ChartShape	Diagramm-Objekt
NotesShape	Objekt für Notizen
HandoutShape	Objekt für Handzettel

Mit `getShapeType()` kann der Typ eines bestimmten Shapes ermittelt werden. Die Größe und Position kann mit `getSize()` und `getPosition()` ermittelt werden, es wird an der Stelle `setSize(Größe)` und `setPosition(Punkt)` erstellt.

7.4.1 Quadrat und Rechteck

Ein Rechteck wird als `com.sun.star.drawing.RectangleShape` erstellt und unterstützt alle Services zur Formatierung von Objekten. Dies sind:

Tabelle 7.118 Die Services zur Formatierung von Objekten

Service	Beschreibung
com.sun.star.drawing.FillProperties	Fläche
com.sun.star.drawing.LineProperties	Linie
com.sun.star.drawing.Text	Text
com.sun.star.style.CharacterProperties	Zeichenformatierung
com.sun.star.style.ParagraphProperties	Absatzformatierung
com.sun.star.drawing.ShadowProperties	Schattenformatierung

Darüber hinaus können die Ecken mit `CornerRadius` in 100stel Millimeter abgerundet werden. Diese Objekte verfügen über die Strukturen `Size` und `Position`, mit den Eigenschaften `Width` und `Height`, beziehungsweise `X` und `Y`. Alle vier Angaben werden als Long-Wert in 0,01 mm angegeben. Das nächste Beispiel zeigt, wie man ein Rechteck erstellen kann:

```
Sub Rechteck

   Dim objDoc As Object
```

```
   Dim objSeite As Object
   Dim objRechteck As Object
   Dim Punkt As New com.sun.star.awt.Point
   Dim Groesse As New com.sun.star.awt.Size
   objDoc = ThisComponent
   objSeite = objDoc.drawPages(0)
   Punkt.x = 10000
   Punkt.y = 5000
   Groesse.Width = 10000
   Groesse.Height = 10000
   objRechteck = _
      objDoc.createInstance("com.sun.star.drawing.RectangleShape")
   objRechteck.Size = Groesse
   objRechteck.Position = Punkt
   objRechteck.CornerRadius = 1000
   objSeite.add(objRechteck)
End Sub
```

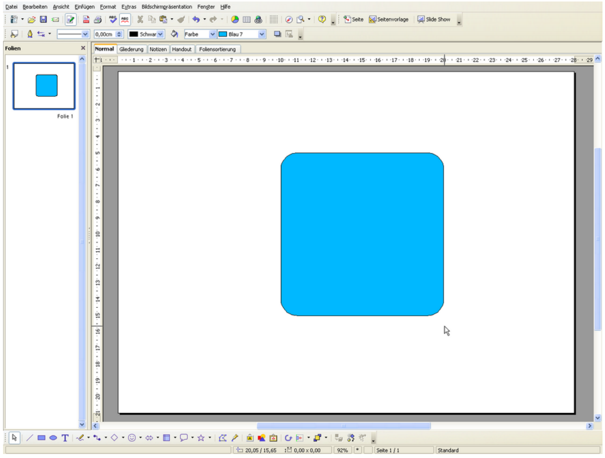


Abbildung 7.144 Das Rechteck

Der Ursprungspunkt berechnet sich dabei als linke, obere Koordinate ausgehend vom Zeichenblatt. Von diesem Punkt aus wird die Breite und Höhe gemessen.

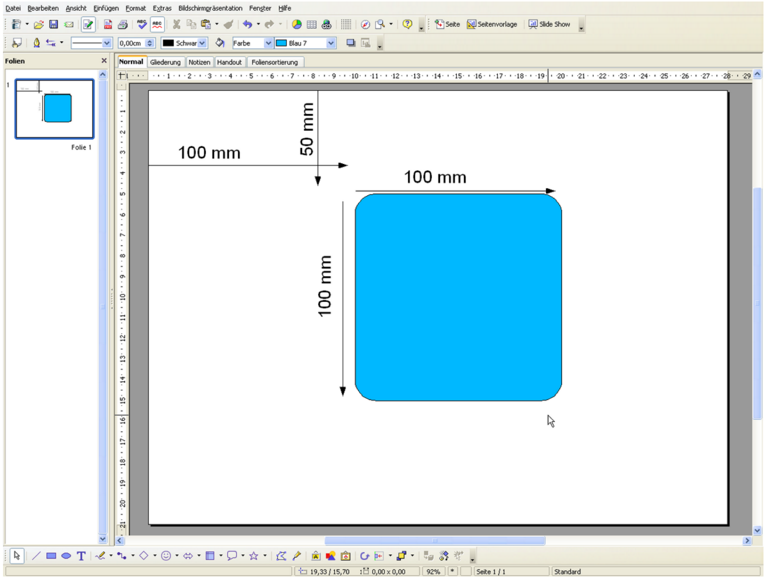


Abbildung 7.145 Die Maße des Rechtecks

7.4.2 Kreis und Ellipse

Für Kreise und Ellipsen ist der Service `com.sun.star.drawing.EllipseShape` zuständig. Kreise und Ellipsen unterstützen ebenso die Services für Flächen, Linien, Texte, Zeichen, Absätze und Schatten. Darüber hinaus verfügen sie über drei weitere Eigenschaften:

Tabelle 7.119 Weitere Eigenschaften der Ellipsen

Eigenschaft	Wert	Beschreibung
CircleKind	Enum	Art des Kreises bzw. der Ellipse (Vorgabewerte gemäß com.sun.star.drawing.CircleKind)
CircleStartAngle	Long	Startwinkel in 10tel Grad (nur bei Kreis- bzw. Ellipsenausschnitten)
CircleEndAngle	Long	Endwinkel in 10tel Grad (nur bei Kreis- bzw. Ellipsenausschnitten)

Für weitere Vielfalt sorgt die Eigenschaft `CircleKind`. Sie bestimmt, ob es sich um eine Vollellipse, einen Ellipsenausschnitt oder einen Ellipsenabschnitt handelt.

Tabelle 7.120 Folgende Werte sind für `CircleKind` des Services com.sun.star möglich:

Wert	Beschreibung
FULL	Vollkreis oder volle Ellipse
CUT	Kreisabschnitt
SECTION	„Tortenstück“
ARC	Winkel ohne Kreislinie

Das folgende Beispiel (*Beispiel02.sxi*) erzeugt mehrere Kreisausschnitte. Der Benutzer kann beliebig viele Zahlen eingeben, aus denen der prozentuale Anteil des Gesamtkreises berechnet wird. Anschließend wird mit einer Zufallsvariablen jedes Segment eingefärbt:

```
Sub Torte

   Dim objDoc As Object
   Dim objSeite As Object
   Dim objTortenstueck As Object
   Dim Punkt As New com.sun.star.awt.Point
   Dim Groesse As New com.sun.star.awt.Size
   Dim intAnzahl As Integer
   Dim dblSumme As Double
   Dim dblWinkel As Double
   Dim dblWinkelEnde As Double
   Dim dblWerte()

   intAnzahl = 0
   ReDim dblWerte(intAnzahl)
   dblWerte(intAnzahl) = 0
   intAnzahl = 1
   ReDim Preserve dblWerte(intAnzahl)
   dblWerte(intAnzahl) = InputBox("Wie lautete der erste Wert?")
   dblSumme = dblWerte(intAnzahl)
```

Der erste Wert wird als Vorgabe auf 0 gesetzt, der Benutzer muss mindestens eine Zahl eingeben. Mit einer Do … Loop-Schleife kann er beliebig viele weitere Werte eingeben, die in einem Datenfeld gespeichert werden:

```
Do Until MsgBox _
  ("Möchten Sie weitere Werte für die Kreissegmente eingeben?", 4) = 7
   intAnzahl = intAnzahl + 1
   Redim Preserve dblWerte(intAnzahl)

   dblWerte(intAnzahl) = _
      InputBox("Wie lautete der " & intAnzahl & ". Wert?")
   dblSumme = dblSumme + dblWerte(intAnzahl)
Loop
objDoc = ThisComponent
objSeite = objDoc.drawPages(0)

Punkt.x = 1000
Punkt.y = 1000
Groesse.Width = 10000
Groesse.Height = 10000
```

Um den Kreis beziehungsweise das Kreissegment muss man sich ein umschriebenes Quadrat denken, dessen linke, obere Ecke in (1000/1000) liegt und das eine Größe von 10 cm × 10 cm hat.

Das Datenfeld wird ausgelesen, der Winkel berechnet. Da der Endwinkel des vorherigen Segmentes gleich dem Anfangswinkel des nächsten Tortenstückes ist, kann der gleiche Wert („dblWinkelEnde") verwendet werden. Da die Winkel „weiterzählen", muss jeweils zu einem Tortenstückwinkel der bereits berechnete dazuaddiert werden:

```
dblWinkelEnde = 0
For intAnzahl = 0 To UBound(dblWerte()) - 1
   objTortenstueck = _
      objDoc.createInstance("com.sun.star.drawing.EllipseShape")
   objTortenstueck.Size = Groesse
   objTortenstueck.Position = Punkt
   objTortenstueck.CircleStartAngle = dblWinkelEnde
   dblWinkel = dblWerte(intAnzahl + 1)
   dblWinkel = dblWinkel /  dblSumme
   dblWinkel = dblWinkel * 36000
   dblWinkelEnde = dblWinkelEnde + dblWinkel
```

```
    objTortenstueck.CircleEndAngle = dblWinkelEnde
```

Jedes Tortenstück ist ein Segment („SECTION“) und wird mit einer zufälligen Farbe gefüllt. Danach wird der Wert auf das Tortenstück „geschrieben“. Natürlich könnte man ihn „formatieren“ oder weitere Texte hinzufügen (die Parteien des Parlamentes oder Bundestages …)

```
    objTortenstueck.CircleKind = _
       com.sun.star.drawing.CircleKind.SECTION
    objTortenstueck.FillColor = RGB(RND * 255, RND * 255, RND * 255)
    objSeite.add(objTortenstueck)
    objTortenstueck.String = dblWerte(intAnzahl + 1)
  Next intAnzahl

End Sub
```

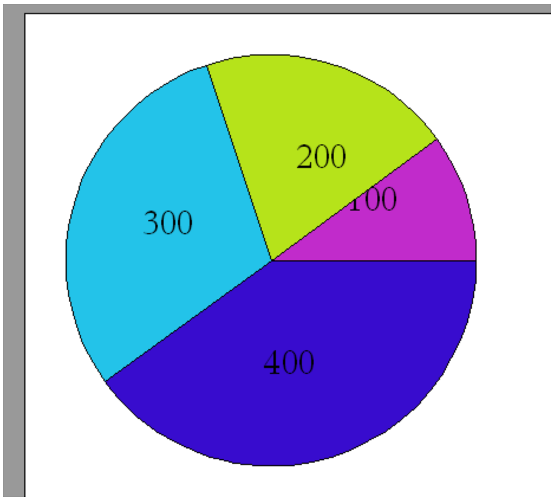


Abbildung 7.146 Eine Torte mit vier (beliebigen) Tortenstücken (SECTIONS)

Beachten Sie, dass der Text erst hinzugefügt werden kann, nachdem das Tortenstück, die Ellipse, beziehungsweise das Rechteck erstellt wurde. Die übrigen Eigenschaften (`Size`, `Width`, `FillColor` …) können vor oder nach der Methode `Add` zugewiesen werden. Besser lesbar ist sicherlich:

```
objTortenstueck = _
   objDoc.createInstance("com.sun.star.drawing.EllipseShape")
```

```
objSeite.add(objTortenstueck)
With objTortenstueck
   .CircleStartAngle = dblWinkelEnde
   .CircleEndAngle = dblWinkelEnde
   .CircleKind = com.sun.star.drawing.CircleKind.SECTION
   :String = "Die grauen Panther"
...
```

7.4.3 Linien

Für Linienobjekte hält OOo den Service `com.sun.star.drawing.LineShape` bereit. Linienobjekte unterstützen alle generellen Formatierungsdienste mit Ausnahme des Dienstes für Flächen:

Tabelle 7.121 Die Services zur Formatierung von Linien:

Service	Beschreibung
com.sun.star.drawing.LineProperties	Linie
com.sun.star.drawing.Text	Text
com.sun.star.style.CharacterProperties	Zeichenformatierung
com.sun.star.style.ParagraphProperties	Absatzformatierung
com.sun.star.drawing.ShadowProperties	Schattenformatierung

Nicht unterstützt werden selbstverständlich die `FillProperties`. Darüber hinausgehende Eigenschaften stellt der Service `LineShape` nicht zur Verfügung.

Linien werden wie Rechtecke oder Ellipsen „gezeichnet“. Man definiert einen Eckpunkt und von ihm ausgehend ein Rechteck, bei dem die Breite und Höhe die Diagonale, also die Linie aufspannen. Das folgende Beispiel (*Beispiel02.sxi*) zeichnet eine vertikale Linie:

```
Punkt.x = 2000
Punkt.y = 2000
Groesse.Width = 0
Groesse.Height = 18000
objLinie = objDoc.createInstance("com.sun.star.drawing.LineShape")
objLinie.Size = Groesse
objLinie.Position = Punkt
```

```
objSeite.add(objLinie)
```

Wenn Sie eine zweite Linie erzeugen möchten, dann müssen Sie nicht nur den Punkt und die „Groesse“ neu definieren, sondern müssen ihn auch mit `createInstance` neu generieren:

```
Punkt.x = 1000
Punkt.y = 19000
Groesse.Width = 26000
Groesse.Height = 0
objLinie = objDoc.createInstance("com.sun.star.drawing.LineShape")
objLinie.Size = Groesse
objLinie.Position = Punkt
objSeite.add(objLinie)
```

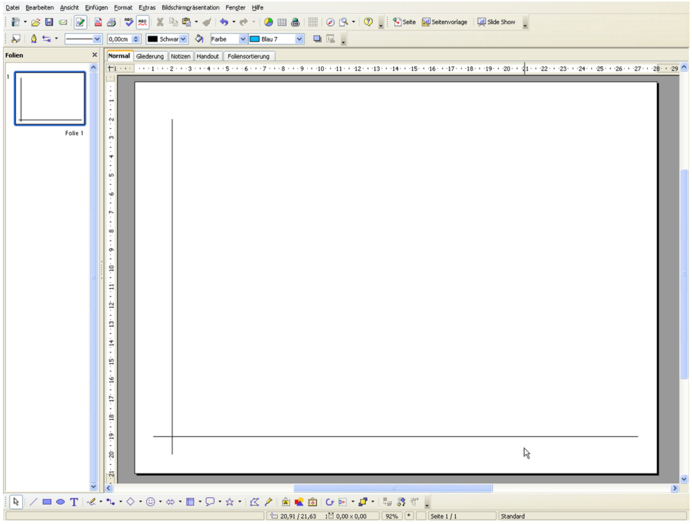


Abbildung 7.147 Eine horizontale und eine vertikale Linie

Damit sind x- und y-Achse eines Diagramms definiert. Nun können am oberen beziehungsweise rechten Ende eine Pfeilspitze eingefügt werden:

```
Punkt.x = 2000
Punkt.y = 2000
Groesse.Width = -500
Groesse.Height = 500
```

```
objLinie = objDoc.createInstance("com.sun.star.drawing.LineShape")
objLinie.Size = Groesse
objLinie.Position = Punkt
objSeite.add(objLinie)

Punkt.x = 2000
Punkt.y = 2000
Groesse.Width = 500
Groesse.Height = 500
objLinie = objDoc.createInstance("com.sun.star.drawing.LineShape")
objLinie.Size = Groesse
objLinie.Position = Punkt
objSeite.add(objLinie)
```

Analog natürlich die zweite Pfeilspitze.

Mit Hilfe einer Schleife werden die beiden Achsen mit Strichen und Beschriftungen versehen:

```
For intZaehler = 1 To 16
   Punkt.x = 1750
   Punkt.y = intZaehler * 1000 + 2000
   Groesse.Width = 500
   Groesse.Height = 0
   objLinie = objDoc.createInstance("com.sun.star.drawing.LineShape")
   objLinie.Size = Groesse
   objLinie.Position = Punkt
   objSeite.add(objLinie)

   Punkt.x = 1200
   Punkt.y = intZaehler * 1000 + 2000
   Groesse.Width = 0
   Groesse.Height = 0
   objLinie = objDoc.createInstance("com.sun.star.drawing.LineShape")
   objLinie.Size = Groesse
```

```
    objLinie.Position = Punkt
    objSeite.add(objLinie)

    objLinie.String = 17 - intZaehler
    objLinie.CharFontName = "Arial"
    objLinie.CharHeight = 12
Next
```

Man hätte die Beschriftung auch direkt auf die Striche schreiben können – dann hätte man allerdings die Position zum Objekt verschieben müssen. Mit einem zweiten Linienshape, das keine Ausdehnung hat, ist dies etwas leichter zu erledigen. Ein zweiter Zähler beschriftet analog die x-Achse:

Der Benutzer wird nach vier Angaben bezüglich der x- und y-Koordinaten aufgefordert (ein Dialog wäre hier schöner!). Um diese vier Punkte wird jeweils ein roter Kreis gezeichnet:

```
x1 = InputBox("Bitte den ersten x-Wert eingeben!")
y1 = InputBox("Bitte den ersten y-Wert eingeben!")
x2 = InputBox("Bitte den zweiten x-Wert eingeben!")
y2 = InputBox("Bitte den zweiten y-Wert eingeben!")
Punkt.x =  2000 + x1 * 1000 - 500
Punkt.y =  2000 + (17 - y1) * 1000 - 500
Groesse.Width = 1000
Groesse.Height = 1000
   objLinie = _
      objDoc.createInstance("com.sun.star.drawing.EllipseShape")
   objLinie.Size = Groesse
   objLinie.Position = Punkt
objSeite.add(objLinie)
objLinie.FillColor = RGB(255, 255, 255)
objLinie.LineColor = RGB(255, 0, 0)
Punkt.x =  2000 + x2 * 1000 - 500
Punkt.y =  2000 + (17 - y2) * 1000 - 500
Groesse.Width = 1000:    Groesse.Height = 1000
   objLinie = _
      objDoc.createInstance("com.sun.star.drawing.EllipseShape")
```

```
        objLinie.Size = Groesse
        objLinie.Position = Punkt
    objSeite.add(objLinie)
    objLinie.FillColor = RGB(255, 255, 255)
    objLinie.LineColor = RGB(255, 0, 0)
```

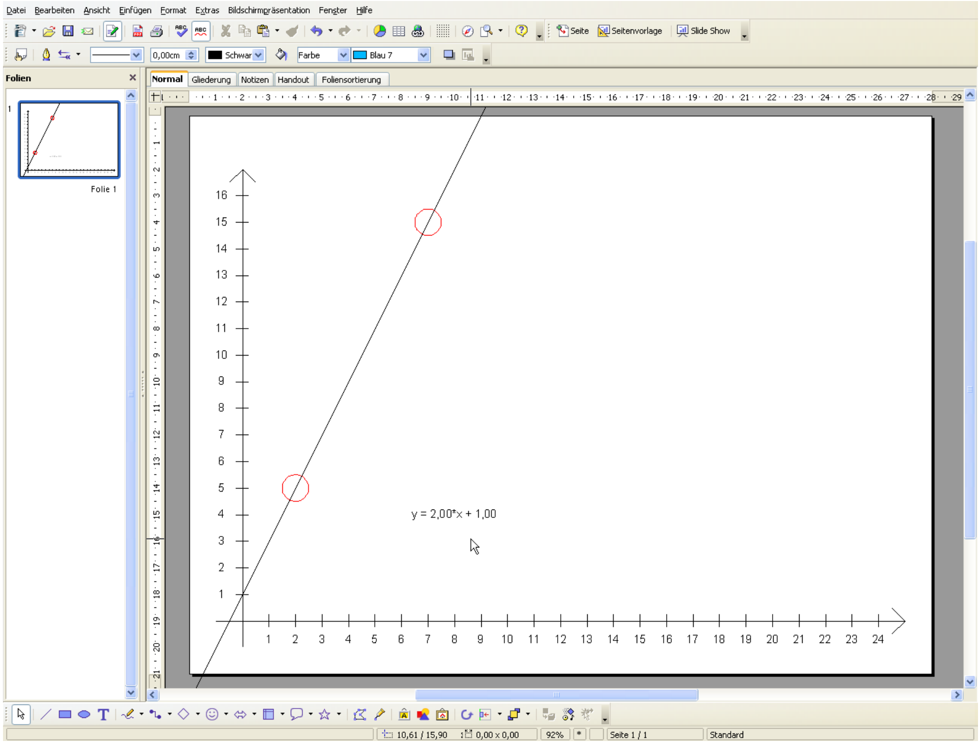


Abbildung 7.148 Das fertige Liniendiagramm

Anschließend werden der y-Achsenabschnitt und die Steigung berechnet, die Grade durch die beiden Punkte erzeugt und die Formel auf das Zeichenblatt geschrieben:

```
    m = (y1 - y2) / (x1 - x2)
    b = y1 - m * x1

    Punkt.x = 10000
    Punkt.y = 15000
```

```
Groesse.Width = 0
Groesse.Height = 0
objLinie = objDoc.createInstance("com.sun.star.drawing.LineShape")
objLinie.Size = Groesse
objLinie.Position = Punkt
objSeite.add(objLinie)

objLinie.String = "y = " & Format(m, "0.00") & "*x " & _
   IIf(b < 0, "", "+ ") & Format(b, "0.00")
objLinie.CharFontName = "Arial"
objLinie.CharHeight = 12
' -- die Funktion

Punkt.x = 0
Punkt.y = 2000 + (17 - (m * (-2) + b)) * 1000
Groesse.Width = 28000
Groesse.Height = m * (-28000)
objLinie = objDoc.createInstance("com.sun.star.drawing.LineShape")
objLinie.Size = Groesse
objLinie.Position = Punkt
objSeite.add(objLinie)
```

Die anderen beiden Fälle einer horizontalen oder vertikalen Geraden (y1 = y2 oder x1 = x2) können gesondert abgehandelt werden. Das Ergebnis sieht dann wie in Abbildung 7.8 aus:

7.4.4 Vielecke

Für Vielecke ist der Service `com.sun.star.drawing.PolyPolygonShape` zuständig. Ein solches Objekt kann beliebig viele Eckpunkte haben, sogar aus mehreren, unabhängigen Punktmengen bestehen.

Für Polygone stehen die gleichen Formatierungsattribute zur Verfügung wie für Rechtecke. Wie in Kapitel 7.3 gezeigt wurde, besitzt der Service `PolyPolygonShape` eine zusätzliche Eigenschaft, über die sich die Koordinaten des Objektes definieren lassen: `PolyPolygon`, das Array, das die Punkte vom Typ `com.sun.star.awt.Point` speichert. Damit werden die „Lage“ und „Größe“ des Polygons (anders als bei Rechtecken und Ellipsen) überflüssig.

Der Streckenzug eines Polygons muss keine zusammenhängende Linie sein – es können auch mehrere (zusammenhängende) Linien ein Vieleck aufspannen. Dabei stellt das „innere“ Objekt ein Loch dar; überschneiden sich zwei Objekte, dann ist die „Schnittmenge“ herausgestanzt. Das folgende Beispiel erstellt zwei gleichseitige Dreiecke, die um jeweils 180 Grad gedreht sind und deren Schnittmenge ein Loch bildet. Um gleichseitige Dreiecke zu erhalten, musste mit der Wurzel-Funktion `sqr` gearbeitet werden (*Beispiel02.sxi*):

```
Sub Poly1
   Dim objDoc As Object
   Dim objSeite As Object
   Dim objStern As Object
   Dim Pkt1(2) As New com.sun.star.awt.Point
   Dim Pkt2(2) As New com.sun.star.awt.Point

   objDoc = ThisComponent
   objSeite = objDoc.drawPages(0)
   objStern = _
      objDoc.createInstance("com.sun.star.drawing.PolyPolygonShape")
   objSeite.add(objStern)
   Pkt1(0).x = 10000
   Pkt1(1).x = 10000 - 5000 / sqr(3)
   Pkt1(2).x = 10000 + 5000 / sqr(3)
   Pkt1(0).y =  5000
   Pkt1(1).y = 10000
   Pkt1(2).y = 10000

   Pkt2(0).x = 10000
   Pkt2(1).x = 10000 - 5000 / sqr(3)
   Pkt2(2).x = 10000 + 5000 / sqr(3)
   Pkt2(0).y = 10000 + 5000 / 3
   Pkt2(1).y = 10000 - 10000 / 3
   Pkt2(2).y = 10000 - 10000 / 3
   objStern.PolyPolygon = Array(Pkt1(), Pkt2())
End Sub
```

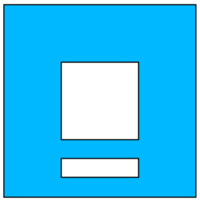


Abbildung 7.149 Ein Objekt mit zwei Löchern

7.4.5 Grafiken

Grafiken werden wie Rechtecke mit Hilfe des Services `com.sun.star.drawing.GraphicObjectShape` eingefügt und unterstützen die Text- und Schattenattribute.

Tabelle 7.122 Grafiken verfügen über weitere Attribute

Eigenschaft	Wert	Beschreibung
GraphicURL	String	URL
AdjustLuminance	Short	Luminanz (Leuchtkraft) der Farben in Prozent (auch negative Werte)
AdjustContrast	Short	Kontrast (auch negative Werte)
AdjustRed AdjustGreen AdjustBlue	Short	die jeweiligen Anteile von Rot, Grün und Blau (auch negative Werte)
Gamma	Short	der Gamma-Wert der Grafik
Transparency	Short	Transparenz in Prozent
GraphicColorMode	Enum	Farbmodus, z.B. Standard, Graustufen, Schwarz-Weiß (Vorgabewert gemäß com.sun.star.drawing.ColorMode)

Damit kann man die verschiedenen Farbmodi schnell darstellen (*Beispiel02.sxi*):

```
objDoc = ThisComponent
objSeite = objDoc.drawPages(0)
strPfad = objDoc.URL
strPfad = Left(strPfad, Len(strPfad) - 14)
strPfad = strPfad & "PIC00017.jpg"
```

```
For i = 0 To 4
   For j = 0 To 4
      Punkt.x = 2000 + 5000 * i
      Punkt.y = 1000 + 4000 * j
      Groesse.Width = 4000:          Groesse.Height = 3000
      objGrafik = _
      objDoc.createInstance("com.sun.star.drawing.GraphicObjectShape")
      objGrafik.Size = Groesse
      objGrafik.Position = Punkt
      objGrafik.GraphicURL = ConvertToURL(strPfad)
      objSeite.add(objGrafik)
      objGrafik.AdjustContrast = -80 + i * 40
      objGrafik.AdjustLuminance = -80 + j * 40
      objGrafik.GraphicColorMode = _
         com.sun.star.drawing.ColorMode.STANDARD
   Next j
Next i
```

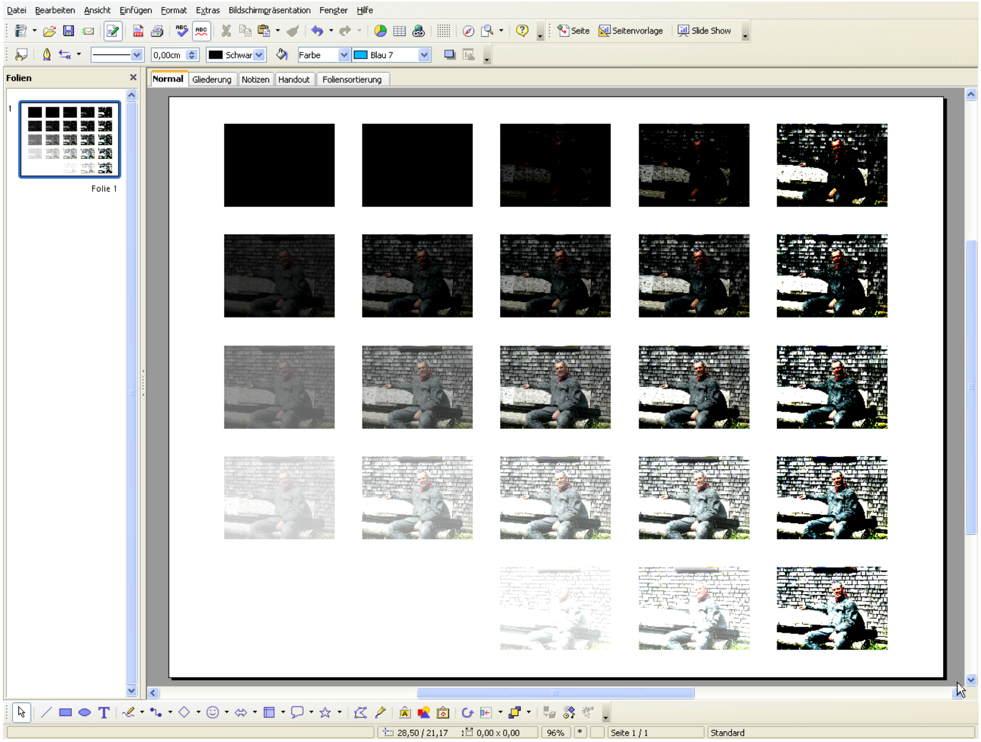


Abbildung 7.150 Ein Bild in verschiedenen Kontrasten und Helligkeiten

Oder statt den Kontrast und die Helligkeit zu „verschieben“, kann man auch die Anteile der Farben Rot, Grün und Gelb (oder nur zwei davon) modifizieren:

```
objGrafik.AdjustBlue = -80 + i * 40
objGrafik.AdjustGreen = -80 + j * 40
objGrafik.AdjustRed = -80 + j * 40
```

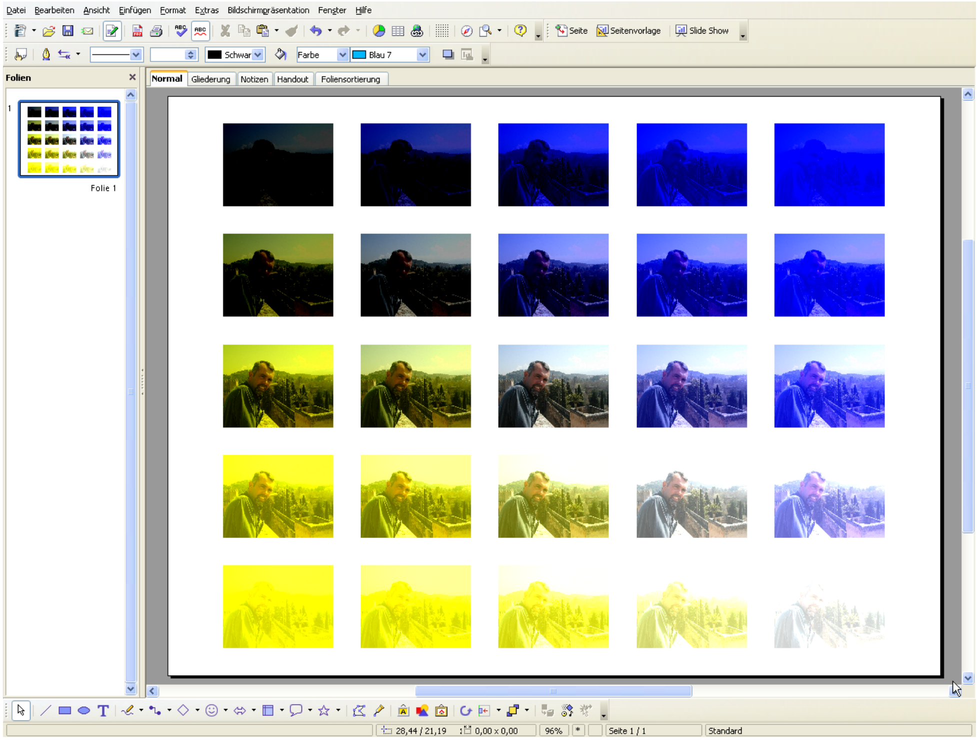


Abbildung 7.151 Ein Bild in verschiedenen Farben

7.5 Flächenattribute

Die Flächenattribute sind in dem Service `com.sun.star.drawing.FillProperties` zusammengefasst. In OOo existieren vier verschiedene Arte, eine Fläche zu formatieren:

- Einfarbige Flächen
- Farbverläufe
- Schraffuren
- Bitmap-Muster

7.5.1 Einfarbige Flächen

Der Füllmodus eines Objektes wird über die Eigenschaft `FillStyle` festgelegt. Die Werte sind in `com.sun.star.drawing.FillStyle` definiert. Die wichtigste Eigenschaft ist sicherlich FillColor, die einen Long-Wert erhält. Darüber hinaus kann der `FillStyle` auf `SOLID` gesetzt werden.

Zwar können auch Bitmap-Muster, Schraffuren und Verläufe halbtransparent dargestellt werden, aber bei vollflächigen Objekten macht die Eigenschaft `FillTransparence` sicherlich am meisten Sinn. Ihr wird ein Prozentwert zugewiesen:

```
objRechteck.FillTransparence = 50
```

7.5.2 Farbverlauf

Um einen Verlauf zu erzeugen, muss der `FillStyle` auf `GRADIENT` gesetzt werden. Ein in OOo definierter Farbverlauf wird mit `FillTransparenceGradientName` zugewiesen. Eigene Verläufe werden mit der Struktur „com.sun.star.awt.Gradient“ definiert. Diese Eigenschaft wird definiert, um die Eigenschaft `FillGradient` zuzuweisen (*Beispiel03.sxi*).

Tabelle 7.123 Die Eigenschaft `FillGradient` enthält folgende Attribute:

Eigenschaft	**Wert**	**Beschreibung**
Style	Enum	Gradientenart, zum Beispiel linear oder radial (Vorgabewerte gemäß com.sun.star.awt.GradientStyle)
StartColor	Long	Anfangsfarbe des Farbverlaufs
EndColor	Long	Endfarbe des Farbverlaufs
Angle	Short	Drehwinkel des Farbverlaufs in 10tel Grad
XOffset	Short	x-Koordinate, bei der welcher Farbverlauf beginnt, Angaben in 100stel Millimetern
YOffset	Short	y-Koordinate, bei welcher der Farbverlauf beginnt, Angaben in 100stel Millimetern
StartIntensity	Short	Helligkeit der StartColor in Prozent
EndItensity	Short	Helligkeit der EndColor in Prozent
StepCount	Short	Anzahl der Farbabstufungen

```
   Dim Verlauf As New com.sun.star.awt.Gradient
...
```

```
objSeite.add(objRechteck)
With Verlauf
    .Style = com.sun.star.awt.GradientStyle.LINEAR
    .StartColor = RGB(255,0,0)
    .EndColor = RGB(0,255,255)
    .StartIntensity = 150
    .EndIntensity = 150
    .Angle = 450
    .StepCount = 90
End With
objRechteck.FillStyle = com.sun.star.drawing.FillStyle.GRADIENT
objRechteck.FillGradient = Verlauf
```

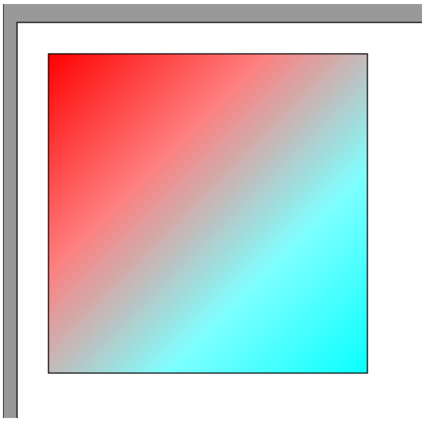


Abbildung 7.152 Rechteck mit Verlauf

Beachten Sie, dass auch hier zuerst das Objekt mit der Methode `add` erstellt werden muss und anschließend der Verlauf zugewiesen wird!

7.5.3 Linienmuster

Um eine Fläche mit einem Linienmuster darzustellen, muss ihrer Eigenschaft `FillStyle` der Wert `HATCH` zugewiesen werden. Der Programmcode zur Definition des Linienmusters ähnelt

stark dem von Farbverläufen. Auch hier kommt eine Hilfstruktur zum Einsatz, die das Aussehen festlegt. Die Struktur für Linienmuster lautet `com.sun.star.drawing.Hatch`

Tabelle 7.124 Das Linienmuster besitzt folgende Attribute:

Eigenschaft	Wert	Beschreibung
Style	Enum	Art des Linienmusters: einfach, kariert oder kariert mit Diagonalen (Vorgabewerte gemäß com.sun.star.awt.HatchStyle)
Color	Long	Die Farbe der Linien
Distance	Long	Der Abstand zwischen den Linien in 100stel Millimetern
Angle	Short	Der Drehwinkel des Linienmusters in 10tel Grad

Das folgende Beispiel (*Beispiel03.sxi*) demonstriert den Einsatz einer Hatch-Struktur:

```
   Dim Muster As New com.sun.star.drawing.Hatch
...
   objSeite.add(objRechteck)
   With Muster
      .Style = com.sun.star.drawing.HatchStyle.SINGLE
      .Color = RGB(255, 0, 0)
      .Distance = 500
      .Angle = 450
   End With
   objRechteck.FillHatch = Muster
```

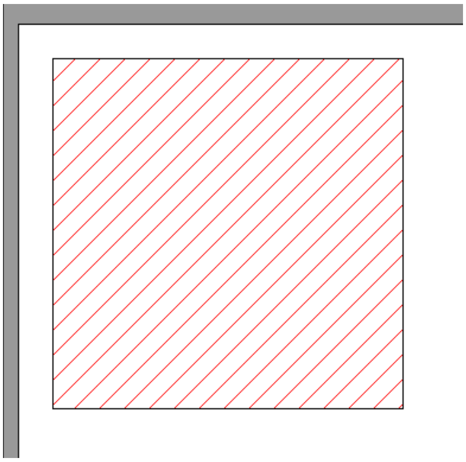


Abbildung 7.153 Rechteck mit Muster

7.5.4 Bitmaps

Um ein Bitmap-Muster auf einem Objekt darzustellen, muss die Eigenschaft `FillStyle` den Wert `BITMAP` besitzen. Ist das Bitmap in StarOffice bereits eingebunden, so genügt es, seinen Namen in die Eigenschaft `FillBitMapName` einzutragen und die gewünschte Darstellungsart (einfach, gekachelt oder gestreckt) über die Eigenschaft `FillBitmapMode` festzulegen (Vorgabewerte gemäß `com.sun.star.drawing.BitmapMode`).

Möchten Sie hingegen mit einer externen Datei arbeiten, so können Sie die zugehörige URL über die Eigenschaft `FillBitmapURL` definieren.

Das folgende Beispiel erzeugt ein Rechteck, auf dessen Fläche das in StarOffice vorhandene Bitmap „Sky“ projiziert wird. Das Bitmap wird dabei in Form von Kacheln neben- und untereinander so oft wiederholt, dass es die komplette Fläche des Rechtecks bedeckt.

```
objRechteck.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
objRechteck.FillBitmapName = "Sky"
objRechteck.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT
```

7.6 Linienattribute

Alle Linien und Objekte, die eine Linie als Umrandung aufweisen, unterstützen den Service `com.sun.star.drawing.LineStyle`.

Tabelle 7.125 `com.sun.star.drawing.LineStyle` bietet folgende Eigenschaften an:

Eigenschaft	Wert	Beschreibung
LineStyle	Enum	Linienart (Vorgabewerte gemäß com.sun.star.drawing.-LineStyle: NONE, SOLID oder DASH)
LineDash	Enum	Punktierung (Vorgabewerte gemäß com.sun.star.drawing.LineDash): Style: Vorgabewert gemäß com.sun.star.drawing.DashStyle: RECT, ROUND, RECTRELATIVE und ROUNDRELATIVE Dots: Anzahl der Punkte (Long) DotLen: Länge der Punkte (Long) Dashes: Anzahl der Striche (Short) DashLen: Länge der Striche (Long) Distance: Abstand zwischen den Punkten als Long
LineColor	Long	die Linienfarbe
LineTransparence	Short	die Linientransparenz
LineWidth	Long	die Linienstärke in 100stel Millimetern
LineJoint	Enum	Übergänge an Verbindungsstellen (Vorgabewerte gemäß com.sun.star.drawing.LineJoint: NONE, MIDDLE, BEVEL, MITER und ROUND)
offene Linien		
LineStartName	Objekt	Name des Anfangspunktes als PolyPolygonBezierCoords
LineStart		Linienbeginn
LineEnd		Linienende
LineStartCenter LineEndCenter	Boolean	Linien zentriert im Linienanfangs- oder Linienenden-Objekt
LineStartWidth LineEndWidth	Long	Breite des Anfangs- oder End-Polygons

Das nächste Beispiel (*Beispiel03.sxi*) erzeugt ein Rechteck mit einer durchgezogenen, halbtransparenten Linie, dessen vier Ecken abgekantet sind:

```
With objRechteck
   .LineColor = RGB(255, 0, 0)
   .LineTransparence = 50
   .LineWidth = 3000
   .LineJoint = com.sun.star.drawing.LineJoint.BEVEL
```

```
    .LineStyle = com.sun.star.drawing.LineStyle.SOLID
End With
```

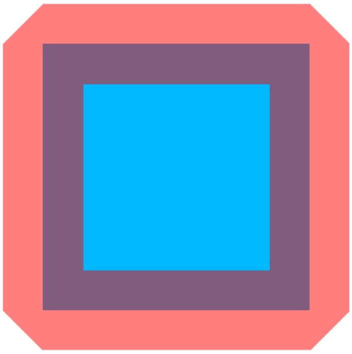


Abbildung 7.154 Rechteck mit halbtransparenten, abgekanteten Ecken

7.7 Textattribute (von Zeichenobjekten)

Neben den in Kapitel 6 beschriebenen Texteigenschaften für die Formatierung von Textzeichen und –absätze stehen folgende Eigenschaften zur Verfügung:

Tabelle 7.126 Weitere Eigenschaften des Services com.sun.star.drawing.TextProperties:

Eigenschaft	Wert	Beschreibung
TextAutoGrowHeight TextAutoGrowWidth	Boolean	passt die Höhe oder Breite des Zeichenelements an den enthaltenen Text an
TextHorizontalAdjust	Enum	horizontale Position des Textes innerhalb des Zeichenelementes (Vorgabewerte gemäß com.sun.star.drawing.TextHorizontalAdjust: LEFT, CENTER, RIGHT und BLOCK)
TextVerticalAdjust	Enum	vertikale Position des Textes innerhalb des Zeichenelementes (Vorgabewerte gemäß com.sun.star.drawing.TextVerticalAdjust: TOP, CENTER, BOTTOM, BLOCK)
TextContourFrame	Boolean	Text wird an den Ecken des Objektes ausgerichtet
TextFitToSize	Enum	Anpassung gemäß com.sun.star.drawing.TextFitTo-

Eigenschaft	Wert	Beschreibung
		SizeType: NONE, PROPORTIONAL, ALLLINES und RESIZEATTR
TextLeftDistance TextRightDistance TextUpperDistance TextLowerDistance	Long	linker, rechter, oberer und unterer Abstand zwischen Zeichenelement und Text in 100stel Millimetern
TextMaximumFrameHeight TextMinimumFrameHeight TextMaximumFrameWidth TextMinimumFrameWidth	Long	Ober- und Untergrenze der Textgröße
IsNumbering	Boolean	Aufzählungszeichen
NumberingRules	Enum	beschreibt die Art der Aufzählung gemäß com.sun.star.style.NumberingRule
TextAnimationAmount	Long	Anzahl der Pixel, die der Text verschoben wird
TextAnimationCount	Long	die Anzahl der Wiederholungen der Textanimation
TextAnimationDelay	Long	Verzögerung in Millisekunden
TextAnimationDirection	Enum	Aufzählung gemäß com.sun.star.drawing.TextAnimationDirection: LEFT, RIGHT, UP und DOWN
TextAnimationKind	Enum	Aufzählung gemäß com.sun.star.drawing.TextAnimationKind: NONE, BLINK, SCROLL, ALTERNATE und SLIDE
TextAnimationStartInside TextAnimationStopInside	Boolean	der Text ist zu Beginn oder am Ende sichtbar
TextWritingMode	Enum	Aufzählung gemäß com.sun.star.text.TextWriting-Mode: LR_TB, RL_TB und TB_RL

Im folgenden Beispiel werden die Eigenschaften eines Rechtecks in ein Rechteck geschrieben (so ähnlich habe ich die Eigenschaften der Zeichnungsobjekte erzeugt):

```
objRechteck.FillColor = RGB(255, 255, 255)
objRechteck.String = objRechteck.dbg_properties
objRechteck.TextFitToSize = _
    com.sun.star.drawing.TextFitToSizeType.PROPORTIONAL
```

Ein weiteres Beispiel zu Schaltflächen und Textfeldern finden Sie in Kapitel 2 bei den Dateien unter „Tischaufsteller“.

7.8 Schattenattribute

Anders als zu den Füllflächen gibt es bei den Schattenobjekten nicht viel zu sagen. Sie unterstützen den Service `com.sun.star.drawing.ShadowProperties` und verfügen über folgende Eigenschaften:

Tabelle 7.127 Die Eigenschaften des Schattenobjektes:

Eigenschaft	**Wert**	**Beschreibung**
Shadow	Boolean	verfügt über einen Schatten
ShadowColor	Long	die Schattenfarbe
ShadowTransparence	Short	die Transparenz des Schattens in Prozent
ShadowXDistance ShadowYDistance	Long	die horizontale und vertikale Verschiebung des Schattens gegenüber des Objekts

7.9 Zeichenobjekte bearbeiten

7.9.1 Objekte gruppieren

Manchmal müssen nicht nur einzelne Zeichenobjekte erstellt, sondern nachträglich modifiziert werden. So kann der Anwender die einzelnen Elemente gruppieren. Dazu wird ein Objekt vom Service „com.sun.star.drawing.ShapeCollection“ erstellt, dem die einzelnen Shapes einer Seite hinzugefügt werden. Anschließend können mit der Methode `group` alle Elemente dieser Sammlung gruppiert werden:

```
Dim objDoc As Object
Dim objSeite As Object
Dim objRechteck As Object
Dim objShapes As Object
Dim i As Integer

objDoc = ThisComponent
objSeite = objDoc.drawPages(0)
objShapes = createUnoService("com.sun.star.drawing.ShapeCollection")
```

```
   For i = 0 To objSeite.getCount() - 1
      objShapes.Add(objSeite.getByIndex(i))
   Next
   objSeite.group(objShapes)
```

Ob ein Objekt gruppiert ist, kann überprüft werden, wenn es den Service `com.sun.star.drawing.GroupShape` unterstützt. Da Gruppen wiederum gruppiert werden können, muss man sich einen Trick einfallen lassen. Leider gibt die Methode ungroup nicht sämtliche Objekte zurück, so dass man nicht rekursiv programmieren kann. Deshalb muss man mit einer Schleife so lange über alle Objekte der Seite iterieren, bis keine Gruppe mehr vorhanden ist:

```
Sub GruppeAufheben
   Dim i As Integer
   Dim blnGruppe As Boolean

   objDoc = ThisComponent
   objSeite = objDoc.drawPages(0)

   blnGruppe = True

   Do
      GruppeAuf
      blnGruppe = False
      For i = 0 To objSeite.getCount() - 1
         If objSeite.getByIndex(i).supportsService _
            ("com.sun.star.drawing.GroupShape") Then
            blnGruppe = True
            Exit For
         End If
      Next
   Loop Until blnGruppe = False

End Sub

Sub GruppeAuf
```

```
    Dim i As Integer

    For i = 0 To objSeite.getCount() - 1
       If objSeite.getByIndex(i).supportsService _
          ("com.sun.star.drawing.GroupShape") Then
           objSeite.ungroup(objSeite.getByIndex(i))
       Next
    Next
End Sub
```

7.9.2 Verschieben

Um ein Objekt auf der Seite zu verschieben, müssen die Eigenschaften X und Y verwendet werden, die an einen Punkt übergeben werden. Das folgende Beispiel verschiebt jedes Objekt auf der Seite um einen cm nach rechts und nach unten:

```
    Dim objDoc As Object
    Dim objSeite As Object
    Dim Punkt As New com.sun.star.awt.Point
    Dim i As Integer

    objDoc = ThisComponent
    objSeite = objDoc.drawPages(0)
    For i = 0 To objSeite.getCount() - 1
       Punkt.X = objSeite.getByIndex(i).getPosition().X + 1000
       Punkt.Y = objSeite.getByIndex(i).getPosition().Y + 1000
       objSeite.getByIndex(i).Position = Punkt
    Next
```

Und so kann man ein Objekt über das Zeichenblatt „fahren“ lassen (*Beispiel03.sxi*):

```
    Dim objDoc As Object
    Dim objSeite As Object
    Dim Punkt As New com.sun.star.awt.Point
    Dim intPosX As Integer
```

```
   Dim i As Integer

   objDoc = ThisComponent
   If objDoc.getDrawPages().hasByName("Zug") = True Then
      objSeite = objDoc.getDrawPages().getByName("Zug")
      intPosX = 100
      Punkt.Y = objSeite.getByIndex(0).getPosition().Y

      For i = 1 To 10000
         Punkt.X = intPosX + i
         objSeite.getByIndex(0).Position = Punkt
      Next
   End If
```

Abbildung 7.155 Der Zug am Ende der Fahrt

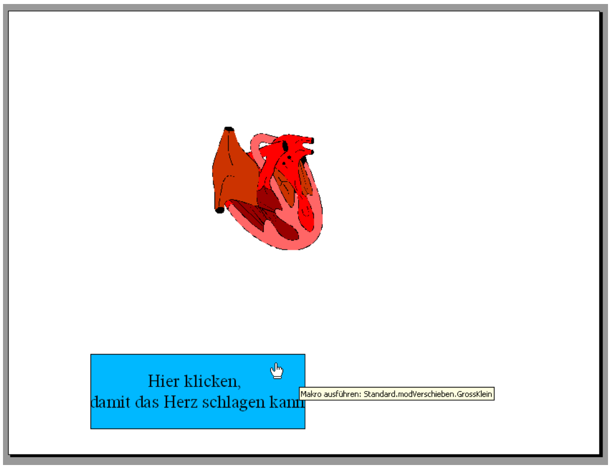
7.9.3 Vergrößern und verkleinern

Mit dem jetzigen Wissen ist auch ein Vergrößern oder Verkleinern nicht mehr schwierig (*Beispiel03.sxi*):

```
Sub GrossKlein
   Dim objDoc As Object
   Dim objSeite As Object
   Dim Groesse As New com.sun.star.awt.Size
```

```
Dim Punkt As New com.sun.star.awt.Point
Dim dblGroesse As Double
Dim intBreite As Integer
Dim intHoehe As Integer
Dim intX As Integer
Dim intY As Integer
Dim intSchlag As Integer

objDoc = ThisComponent
If objDoc.getDrawPages().hasByName("Herzschlag") = True Then
   objSeite = objDoc.getDrawPages().getByName("Herzschlag")
   intHoehe = objSeite.getByIndex(1).Size.Height
   intBreite = objSeite.getByIndex(1).Size.Width

   intX = objSeite.getByIndex(1).getPosition().X
   intY = objSeite.getByIndex(1).getPosition().Y

   For intSchlag = 1 To 5
      For dblGroesse = 1 To 3 Step 0.01
         Groesse.Height = intHoehe * dblGroesse
         Groesse.Width = intBreite * dblGroesse
         Punkt.X = intX - (intBreite * dblGroesse - intBreite) / 2
         Punkt.Y = intY - (intHoehe * dblGroesse - intHoehe) / 2
         objSeite.getByIndex(1).Size = Groesse
         objSeite.getByIndex(1).Position = Punkt
      Next

      For dblGroesse = 3 To 1 Step -0.01
         Groesse.Height = intHoehe * dblGroesse
         Groesse.Width = intBreite * dblGroesse
         Punkt.X = intX - (intBreite * dblGroesse - intBreite) / 2
         Punkt.Y = intY - (intHoehe * dblGroesse - intHoehe) / 2
```

```
            objSeite.getByIndex(1).Size = Groesse
            objSeite.getByIndex(1).Position = Punkt
         Next
      Next
   End If

End Sub
```



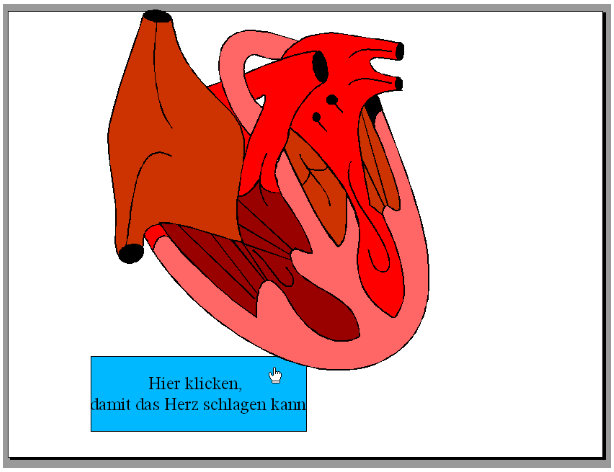


Abbildung 7.156 Das „Objekt“ wird vergrößert und verkleinert.

7.9.4 Kombinieren

Zwar können leicht Streckenzüge mit Löchern erzeugt werden, aber bei komplexen Figuren ist dies schwierig. Abhilfe schafft hier das Kombinieren, das ebenso funktioniert wie das Gruppieren.

```
objShape = objSeite.combine(objShapes)
```

oder auch nur:

```
objSeite.combine(objShapes)
```

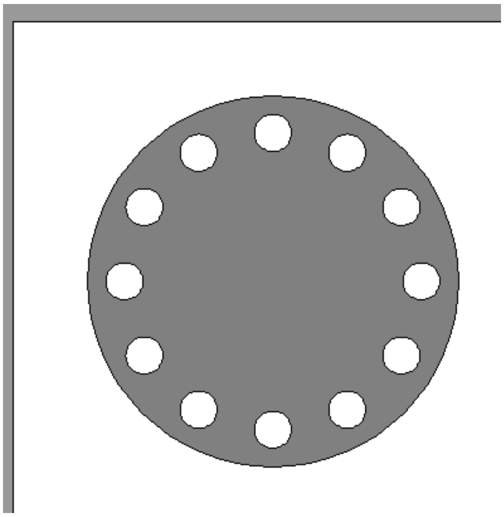


Abbildung 7.157 Kombinierte Objekte

7.9.5 Rotieren und Scheren von Zeichenobjekten

Alle Zeichenobjekte lassen sich drehen und scheren. Der Service `com.sun.star.drawing.RotationDescriptor` unterstützt sie dabei. Er stellt die Eigenschaften `RotateAngle` und `ShearAngle` zur Verfügung, der einen Long-Wert verlangt, der jeweils den Drehwinkel beziehungsweise den Schwerwinkel in 100stel Grad beschreibt:

```
objShape.RotateAngle = 9000
```

Leider werden eingefügte Objekte eines Rechtecks nicht mit gedreht oder geschert. Und es ist auch nicht möglich, ein Objekt zu drehen oder scheren, da sie den Service `com.sun.star.drawing.RotationDescriptor` nicht unterstützen.

Obwohl die beiden Eigenschaften einen Long-Wert entgegennehmen, liefern sie einen Fehler, wenn eine Integer-Zahl überschritten ist! Auch das Scheren ist in vielen Fällen fehleranfällig!

Zwei weitere Beispiele finden Sie in *Beispiel03.sxi* im Modul „modDrehenScheren“ `Drehen` und `Scheren`.

7.9.6 Shapes verbinden

Um zwei Objekte miteinander zu verbinden, wird das Connector-Shape verwendet. Es unterstützt den Service `com.sun.star.drawing.ConnectorShape`. Da die Eigenschaften für den Anfang und für das Ende die gleichen sind, werden sie im Folgenden paarweise aufgelistet:

Tabelle 7.128 Der Verbinder verfügt über folgende Eigenschaften:

Eigenschaft	Wert	Beschreibung
StartShape EndShape	Object/Void	Das Objekt, an das der Verbinder geklebt wird
StartGluePointIndex EndGluePointIndex	Long	Die Nummer des Verbindungspunktes, beginnend bei 0 (oben) bis 3 (links) – im Uhrzeigersinn
StartPosition EndPosition	Object	Die Start- und Endposition, falls der Verbinder nicht an das Objekt geklebt wird.
EdgeLine1Delta EdgeLine2Delta EdgeLine3Delta	Long	Linienversatz
EdgeKind	Enum	Konstante des Typs com.sun.star.drawing.ConnectorType: STANDARD, CURVE, LINE (fehlerhaft!) und LINES

```
...
   For i = 1 To 3
      objVerbinder = _
         objDoc.createInstance("com.sun.star.drawing.ConnectorShape")
```

```
    objSeite.add(objVerbinder)
    objVerbinder.StartShape = objRechteck1
    objVerbinder.StartGluePointIndex = 2
    objVerbinder.EndShape = objRechteck2
    objVerbinder.EndGluePointIndex = 0
    If i = 1 Then
       objVerbinder.EdgeKind = com.sun.star.drawing.ConnectorType.LINES
    ElseIf i = 2 Then
       objVerbinder.EdgeKind = com.sun.star.drawing.ConnectorType.CURVE
    ElseIf i = 3 Then
     objVerbinder.EdgeKind = _
          com.sun.star.drawing.ConnectorType.STANDARD
    End If
Next
```

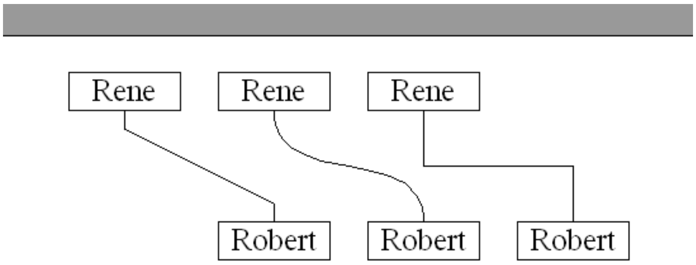


Abbildung 7.158 Drei verschiedene Verbinder

Mit dem Service „com.sun.star.drawing.GluePoint2“ ist es möglich, einen neuen Klebepunkt zu erzeugen, mit dessen Hilfe Linien an andere Punkte befestigt werden können (*Beispiel03.sxi*):

```
objKlebePunkt = createUnoStruct("com.sun.star.drawing.GluePoint2")
objKlebePunkt.IsRelative = False
objKlebePunkt.Escape = com.sun.star.drawing.EscapeDirection.UP
objKlebePunkt.PositionAlignment = _
   com.sun.star.drawing.Alignment.CENTER
objKlebePunkt.IsUserDefined = True
```

```
    objKlebePunkt.Position.X = 4000 * i + 7000
    objKlebePunkt.Position.Y = 5000
    objVerbinder.StartShape = objRechteck1
    objVerbinder.StartGluePointIndex = _
       objRechteck1.getGluePoints().insert(objKlebePunkt)
...
```

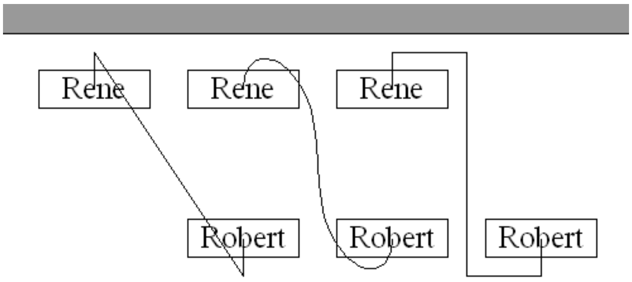


Abbildung 7.159 Drei andere Verbinder

Tabelle 7.129 Die Verbindungspunkte verfügen über folgende Eigenschaften:

Eigenschaft	**Wert**	**Beschreibung**
Position	Object	der Punkt, an dem die Verbinder-Linie befestigt wird
IsRelative	Boolean	wenn der Wert „True“ ist, dann wird der Abstand in 1/100 % angegeben
PositionAlignment	Enum	Wert der Aufzählung com.sun.star.drawing.Alignment: TOP_LEFT, TOP_RIGHT, LEFT, CENTER, RIGHT, BOTTOM_LEFT, BOTTOM und BOTTOM_RIGHT – gibt die Position an
Escape	Enum	Wert der Aufzählung com.sun.star.drawing.EscapeDirection: SMART, LEFT, RIGHT, UP, DOWN, HORIZONTAL und VERTICAL
IsUserDefined	Boolean	ist der Wert „False“, dann ist es ein Standardklebepunkt

Die Linie kann mit einem von OOo vorgegebenen Stil formatiert werden:

```
objVerbinder.Style = objDoc.getStyleFamilies(). _
```

```
getByName("graphics").getByName("objectwitharrow")
```

Man könnte auch selbst einen Stil definieren und anwenden.

Beachten Sie, dass OOo die internen Namen verwendet, also nicht „Präsentationsobjektvorlagen“ und „Grafikobjektvorlagen“, sondern „Standard“ und „graphics“. Auch die Stile verwenden die internen Namen, die man über eine Schleife auslesen lassen kann. Sie lauten: standard, objectwitharrow, objectwithshadow, objectwithoutfill, text, textbody, textbodyjustified, textbodyindent, title, title1, title2, headline, headline1, headline2 und measure.

Natürlich können Sie auch nur eine Pfeilspitze anfügen (selbstverständlich wieder über den englischen Namen):

```
objVerbinder.LineStartName = "Arrow"
```

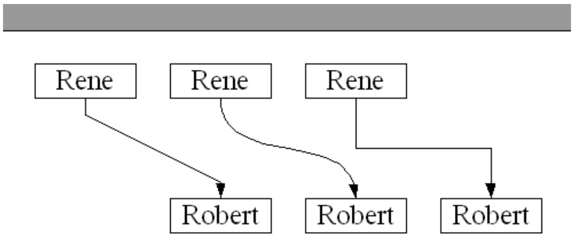


Abbildung 7.160 Drei Verbinder mit Pfeilspitzen

Tabelle 7.130 Die internen Namen der Pfeilspitzen für LineEndName und LineStartName lauten:

deutscher Name	**interner (englischer) Name**
Kein	leere Zeichenkette
Pfeil	Arrow
Quadrat	Square
Schmaler Pfeil	Small Arrow
Bemaßungsenden	Dimension Lines
Doppelpfeil	Double Arrow
Abgerundeter kurzer Pfeil	Rounded short Arrow
Symmetrischer Pfeil	Symmetric Arrow
Linien-Pfeil	Line Arrow
Abgerundeter langer Pfeil	Rounded large Arrow

deutscher Name	interner (englischer) Name
Kreis	Circle
Quadrat 45	Square 45
Pfeil konkav	Arrow concave

Ein Beispiel zum automatischen Generieren von Shapes und Verbindern findet sich in Kapitel 13.

7.9.7 Suchen und Ersetzen

Zeichendokumente bieten ähnlich wie Textdokumente eine Funktion zum Suchen und Ersetzen. Die Technik, die dabei verwendet wird, ist ähnlich wie bei Textdokumenten. Dabei wird der Descriptor jedoch nicht auf das Dokument-Objekt angewandt, sondern auf die entsprechenden Shapes. Das folgende Beispiel (*Beispiel03.sxi*) verdeutlicht einen Ersetzungsvorgang innerhalb eines Zeichenblattes:

```
Dim objDoc As Object
Dim objSeite As Object
Dim objErsetzDescriptor As Object

objDoc = ThisComponent
objSeite = objDoc.drawPages(0)
objErsetzDescriptor = objSeite.createReplaceDescriptor()
objErsetzDescriptor.SearchString = "Rene"
objErsetzDescriptor.ReplaceString = "@@@"
objSeite.ReplaceAll(objErsetzDescriptor)
objErsetzDescriptor.SearchString = "Robert"
objErsetzDescriptor.ReplaceString = "Rene"
objSeite.ReplaceAll(objErsetzDescriptor)
objErsetzDescriptor.SearchString = "@@@"
objErsetzDescriptor.ReplaceString = "Robert"
objSeite.ReplaceAll(objErsetzDescriptor)
```

7.10 Präsentationen

7.10.1 Die Animation (Bildschirmpräsentationseinstellungen)

In OOo können nicht nur Zeichnungen programmiert werden – auch Animationsabläufe können automatisiert werden. Das zuständige Objekt lautet: `Presentation`. Es verfügt über die Methode `start()` und `end()`, mit der eine Präsentation gestartet beziehungsweise beendet werden kann.

Tabelle 7.131 Daneben stellt Presentation einige Eigenschaften zur Verfügung

Eigenschaft	Wert	Beschreibung
AllowAnimations	Boolean	Animationen sind aktiviert
CustomShow	String	Name der „individuellen Bildschirmpräsentation“
FirstPage	String	Name der ersten Seite der Präsentation
IsAlwaysOnTop	Boolean	die Präsentation „liegt oben“, d.h. „im Vordergrund“
IsAutomatic	Boolean	Diawechsel manuell („True“) oder nicht („False“)
IsEndless	Boolean	beginnt automatisch wieder am Anfang
IsFullScreen	Boolean	im gesamten Fenster
IsLivePresentation	Boolean	aktiviert den „Live-Modus“
IsMouseVisible	Boolean	schaltet den Mauszeiger ein oder aus
UsePen	Boolean	Mauszeiger als Stift
IsShowAll	Boolean	???
IsShowLogo	Boolean	blendet Logo ein
IsTransitionOnClick	Boolean	Diawechsel bei Klick auf Hintergrund
Pause	Long	die Dauer des schwarzen Bildschirms, der nach der Präsentation erscheint (in 100stel Sekunden)
StartWithNavigator	Boolean	Navigator sichtbar

Die Präsentation könnte beispielsweise wie folgt eingerichtet und gestartet werden:

```
With objDoc.Presentation
   .IsAutomatic = False
   .AllowAnimations = True
   .IsLivePresentation = False
   .IsShowAll = True
   .IsEndless = False
```

```
    .IsAlwaysOnTop = True
    .FirstPage = "Seite 2"
    .start()
End With
```

Die Eigenschaft „IsAutomatic“ wurde falsch bezeichnet – korrekt müsste es wohl heißen: IsNotAutomatic oder IsManuell.

7.10.2 Die Diawechsel (Seitenübergänge, Folienübergänge)

Tabelle 7.132 Für die Seitenübergänge stehen folgende Eigenschaften zur Verfügung:

Eigenschaft	Wert	Beschreibung
Change	Long	der Wechsel der Seiten: Diawechsel automatisch (1) Diawechsel halbautomatisch (2) Diawechsel manuell (0)
Duration	Long	Diaanzeigezeit
Effect	Long	die Effekte (s. Tabelle 7.19)
Sound	String	„Klang“
Speed	Long	Geschwindigkeit. Konstante gemäß com.sun.star.presentation.AnimationSpeed: SLOW, MEDIUM und FAST

Tabelle 7.133 Folgende Werte stehen für com.sun.star.presentation.FadeEffect zur Verfügung:

Effekt	Konstante
Überblenden	
von links oben	FADE_FROM_UPPERLEFT
von oben	FADE_FROM_TOP
von rechts oben	FADE_FROM_UPPERRIGHT
von rechts	FADE_FROM_RIGHT
von rechts unten	FADE_FROM_LOWERRIGHT
von unten	FADE_FROM_BOTTOM
von links unten	FADE_FROM_LOWERLEFT
von links	FADE_FROM_LEFT
Einfliegen	

Effekt	**Konstante**
von links oben	MOVE_FROM_UPPERLEFT
von oben	MOVE_FROM_TOP
von rechts oben	MOVE_FROM_UPPERRIGHT
von rechts	MOVE_FROM_RIGHT
von rechts unten	MOVE_FROM_LOWERRIGHT
von unten	MOVE_FROM_BOTTOM
von links unten	MOVE_FROM_LOWERLEFT
von links	MOVE_FROM_LEFT
Aufdecken	
nach links oben	UNCOVER_TO_UPPERLEFT
nach oben	UNCOVER_TO_TOP
nach rechts oben	UNCOVER_TO_UPPERRIGHT
nach rechts	UNCOVER_TO_RIGHT
nach rechts unten	UNCOVER_TO_LOWERRIGHT
nach unten	UNCOVER_TO_BOTTOM
nach links unten	UNCOVER_TO_LOWERLEFT
nach links	UNCOVER_TO_LEFT
Blenden	
nach innen	FADE_TO_CENTER
nach außen	FADE_FROM_CENTER
vertikal	VERTICAL_STRIPES
horizontal	HORIZONTAL_STRIPES
im Uhrzeigersinn	CLOCKWISE
gegen den Uhrzeigersinn	COUNTERCLOCKWISE
Öffnen/Schließen	
öffnen vertikal	OPEN_VERTICAL
öffnen horizontal	OPEN_HORIZONTAL
schließen vertikal	CLOSE_VERTICAL
schließen horizontal	CLOSE_HORIZONTAL
Schlangenlinie	
von oben	WAVYLINE_FROM_TOP
von rechts	WAVYLINE_FROM_RIGHT
von unten	WAVYLINE_FROM_BOTTOM

Effekt	Konstante
von links	WAVYLINE_FROM_LEFT
Spirale	
nach innen im Uhrzeigersinn	SPIRALIN_LEFT
nach außen im Uhrzeigersinn	SPIRALOUT_LEFT
nach innen gegen den Uhrzeigersinn	SPIRALIN_RIGHT
nach außen gegen den Uhrzeigersinn	SPIRALIN_RIGHT
Rollen	
von oben	ROLL_FROM_TOP
von rechts	ROLL_FROM_RIGHT
von unten	ROLL_FROM_BOTTOM
von links	ROLL_FROM_LEFT
Aufbauen	
von oben	STRETCH_FROM_TOP
von rechts	STRETCH_FROM_RIGHT
von unten	STRETCH_FROM_BOTTOM
von links	STRETCH_FROM_LEFT
Sonstige	
Linien vertikal	VERTICAL_LINES
Linien horizontal	HORIZONTAL_LINES
Auflösen	DISSOLVE
Schachbrett vertikal	VERTICAL_CHECKERBOARD
Schachbrett horizontal	HORIZONTAL_CHECKERBOARD
automatisch (Zufall)	RANDOM
kein Effekt	NONE

So könnten Seitenübergänge programmiert werden:

```
For i = 2 To 6
   objBlatt = objDoc.getDrawPages().insertNewByIndex(i)
   objBlatt.setName("Seite " & i)
   objText = objDoc.createInstance("com.sun.star.drawing.TextShape")
   objBlatt.add(objText)
   Punkt.x = 10000: Punkt.y = 5000
   Groesse.Width = 10000: Groesse.Height = 10000
   objText.Size = Groesse: objText.Position = Punkt
```

```
    objText.String = "Seite " & i
    objBlatt.Effect = _
       com.sun.star.presentation.FadeEffect.MOVE_FROM_LEFT
    objBlatt.Change = 1
    objBlatt.Duration = 1
    objBlatt.Speed = com.sun.star.presentation.AnimationSpeed.FAST
Next i
```

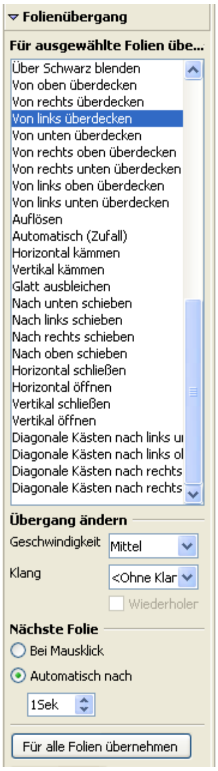


Abbildung 7.161 Die Folienübergänge

7.10.3 Animationseffekte der Shapes (benutzerdefinierte Animationen)

Um die Shapes zu animieren, steht ebenfalls eine Reihe von Eigenschaften zur Verfügung:

Tabelle 7.134 Eigenschaften des Services com.sun.star.presentation.Shape

Eigenschaft	**Wert**	**Beschreibung**
Bookmark	String	die URL der Datei, auf die bei einem Klick weitergeleitet wird
DimColor	Long	Objekt mit Farbe ausblenden
DimHide	Boolean	Objekt unsichtbar machen
DimPrevious	Boolean	ausblenden nach dem Durchführen des Effektes
Effect	Long	Konstante wie in Tabelle 7.19 beschrieben
OnClick	Long	Konstante wie in Tabelle 7.22 aufgelistet
FullPlay	Boolean	Klang wird in voller Lautstärke abgespielt
Sound	String	Klang
SoundOn	Boolean	Klang abspielen
Speed	Long	Geschwindigkeit. Konstante gemäß com.sun.star.presentation.AnimationSpeed: SLOW, MEDIUM und FAST
TextEffect	Long	Konstante wie in Tabelle 7.21 aufgelistet
Verb	Long	OLE-Verbindung

Tabelle 7.135 Folgende Werte stehen für com.sun.star.presentation.AnimationEffect zur Verfügung:

Effekt	**Konstante**
kein Effekt	NONE
Überblenden	
von links oben	FADE_FROM_UPPERLEFT
von oben	FADE_FROM_TOP
von rechts oben	FADE_FROM_UPPERRIGHT
von rechts	FADE_FROM_RIGHT
von rechts unten	FADE_FROM_LOWERRIGHT
von unten	FADE_FROM_BOTTOM
von links unten	FADE_FROM_LOWERLEFT
von links	FADE_FROM_LEFT

Effekt	Konstante
Einfliegen	
von links oben	MOVE_FROM_UPPERLEFT
von oben	MOVE_FROM_TOP
von rechts oben	MOVE_FROM_UPPERRIGHT
von rechts	MOVE_FROM_RIGHT
von rechts unten	MOVE_FROM_LOWERRIGHT
von unten	MOVE_FROM_BOTTOM
von links unten	MOVE_FROM_LOWERLEFT
von links	MOVE_FROM_LEFT
Verkürzt einfliegen	
von links oben	MOVE_SHORT_FROM_UPPERLEFT
von oben	MOVE_SHORT_FROM_TOP
von rechts oben	MOVE_SHORT_FROM_UPPERRIGHT
von rechts	MOVE_SHORT_FROM_RIGHT
von rechts unten	MOVE_SHORT_FROM_LOWERRIGHT
von unten	MOVE_SHORT_FROM_BOTTOM
von links unten	MOVE_SHORT_FROM_LOWERLEFT
von links	MOVE_SHORT_FROM_LEFT
Dehnen	
von links oben	STRETCH_FROM_UPPERLEFT
von oben	STRETCH_FROM_TOP
von rechts oben	STRETCH_FROM_UPPERRIGHT
von rechts	STRETCH_FROM_RIGHT
von rechts unten	STRETCH_FROM_LOWERRIGHT
von unten	STRETCH_FROM_BOTTOM
von links unten	STRETCH_FROM_LOWERLEFT
von links	STRETCH_FROM_LEFT
Dehnen und drehen	
dehnen vertikal	OPEN_VERTICAL
dehnen horizontal	OPEN_HORIZONTAL
drehen vertikal	CLOSE_VERTICAL
drehen horizontal	CLOSE_HORIZONTAL
Blenden	

Effekt	**Konstante**
nach innen	ZOOM_IN
nach außen	ZOOM_OUT
vertikal	VERTICAL_STRIPES
horizontal	HORIZONTAL_STRIPES
im Uhrzeigersinn	CLOCKWISE
gegen den Uhrzeigersinn	COUNTERCLOCKWISE
Öffnen / Schließen	
öffnen vertikal	OPEN_VERTICAL
öffnen horizontal	OPEN_HORIZONTAL
schließen vertikal	CLOSE_VERTICAL
schließen horizontal	CLOSE_HORIZONTAL
Schlangenlinie	
von oben	WAVYLINE_FROM_TOP
von rechts	WAVYLINE_FROM_RIGHT
von unten	WAVYLINE_FROM_BOTTOM
von links	WAVYLINE_FROM_LEFT
Spirale	
nach innen im Uhrzeigersinn	SPIRALIN_LEFT
nach außen im Uhrzeigersinn	SPIRALOUT_LEFT
nach innen gegen den Uhrzeigersinn	SPIRALIN_RIGHT
nach außen gegen den Uhrzeigersinn	SPIRALIN_RIGHT
Laser	
von links oben	LASER_FROM_UPPERLEFT
von oben	LASER_FROM_TOP
von rechts oben	LASER_FROM_UPPERRIGHT
von rechts	LASER_FROM_RIGHT
von rechts unten	LASER_FROM_LOWERRIGHT
von unten	LASER_FROM_BOTTOM
von links unten	LASER_FROM_LOWERLEFT
von links	LASER_FROM_LEFT
Sonstige	
Linien vertikal	VERTICAL_LINES
Linien horizontal	HORIZONTAL_LINES

Effekt	Konstante
Auflösen	DISSOLVE
Schachbrett vertikal	VERTICAL_CHECKERBOARD
Schachbrett horizontal	HORIZONTAL_CHECKERBOARD
automatisch (Zufall)	RANDOM
Erscheinen	APPEAR
an Kurve entlang	PATH

So könnte ein Effekt dargestellt werden:

```
objText.TextEffect = _
   com.sun.star.presentation.AnimationEffect.LASER_FROM_LEFT
```

Tabelle 7.136 Folgende Werte stehen für com.sun.star.presentation.ClickAction (Interaktion) zur Verfügung:

Effekt	Konstante
keine Aktion	NONE
Sprung zur vorhergehenden Seite	PREVPAGE
Sprung zur nächsten Seite	NEXTPAGE
Sprung zur ersten Seite	FIRSTPAGE
Sprung zur letzten Seite	LASTVPAGE
Sprung zu Seite oder Objekt	BOOKMARK
Sprung zu Dokument	DOCUMENT
Objekt unsichtbar machen	INVISIBLE
Klang abspielen	SOUND
Objekt ausblenden	VANISH
Programm ausführen	PROGRAM
Makro ausführen	MACRO
VERB	OLE-Verbindung
Präsentation beenden	STOPPRESENTATION

Draw und Impress (Präsentation und Zeichnung)

Mit der Version 2.0 ermöglichte OOo nicht nur einen Datenbankzugriff, sondern führte ein Datenbankobjekt ein, in dem der Benutzer eigene Datenbanken anlegen und verändern kann. Gleichzeitig ist es möglich, auf bestehende Datenbanken unterschiedlicher Formate zuzugreifen.

8 Datenbankzugriff

8.1 Die StarDatabase Connectivity

OOo verfügt bereits seit der Version 1.1 (StarOffice 7.0) über eine integrierte Schnittstelle zu Datenbanken: die StarDatabase Connectivity (SDBC). Damit ist es möglich, auf unterschiedliche Datenquellen zuzugreifen, unabhängig von den darunter liegenden Datenbanken-Backends. Als Wurzelobjekt für den Zugriff dient das Objekt `DatabaseContext`. Die komplette API der einzelnen Services und Interfaces findet sich im Paket „com.sun.star.sdb“. Zwei weitere Pakete („sdbc“ und „sdbcx“) bauen auf ihr auf und erweitern sie.

8.2 Die Abfragesprache SQL

Als Abfragesprache für SDBC dient SQL. Da es verschiedene SQL-Dialekte gibt, besitzt die SDBC-Komponente von OOo einen eigenen SQL-Parser, der kleine Unterschiede überprüft und korrigiert.

Die SQL-Implementation von SDBC orientiert sich am SQL-ANSI-Standard. Microsoft-spezifische Erweiterungen wie das `INNER JOIN`-Konstrukt werden nicht unterstützt. Diese sind durch Standardbefehle zu ersetzen (`INNER JOIN` beispielsweise durch eine entsprechende `WHERE`-Klausel).

8.3 Arten des Datenbankzugriffs

Über DATEI | NEU | DATENBANK kann eine neue, leere Datenbank im OOo XML-Base-Format erstellt werden. Wurde sie registriert, so steht sie im Menü EXTRAS | OPTIONEN | OOO-BASE | DATENBANKEN als Eintrag.

8.4 Datenquellen

Auf diese Datenquellen kann nun zugegriffen werden. Ausgangspunkt für diesen Zugriff bildet ein Datenbank-Kontext-Objekt, das über die Funktion `createUnoService` erzeugt wird. Es ba-

siert auf dem Service „com.sun.star.sdb.DatabaseContext“ und stellt das Wurzelobjekt für alle Datenbankoperationen dar.

Das folgende Beispiel zeigt, wie sich ein Datenbank-Kontext erstellen und zur Ermittlung der Namen aller verfügbaren Datenquellen verwenden lässt:

```
Sub Welche_Datenbanken

   Dim objDatabaseContext As Object
   Dim objNamen
   Dim i As Integer
   Dim strListe As String
   objDatabaseContext = _
      createUnoService("com.sun.star.sdb.DatabaseContext")
   objNamen = objDatabaseContext.getElementNames()
   For i = 0 To UBound(objNamen())
      strListe = strListe & Chr(13) & objNamen(i)
   Next
   MsgBox strListe
End Sub
```

Die wichtigste Eigenschaft des Services `com.sun.star.sdb.DataSource` ist sicherlich `ElementNames` (ein Array).

Tabelle 8.137 com.sun.star.sdb.DataSource stellt folgende Methoden zur Verfügung:

Methode	Wert	Beschreibung
hasElements(void)	Boolean	überprüft, ob registrierte Elemente vorhanden sind
createEnumeration(void)	Object	erstellt ein Aufzählungsobjekt
getByName(String)	Empty	greift auf eine Datenbank zu
getElementNames(void)	Array	greift auf alle registrierten Datenbanken zu
hasByName(String)	Boolean	überprüft, ob eine registrierte Datenbank vorhanden ist
getRegisteredObject(String)	Object	greift auf eine Datenbank zu
registerObject(String, Objekt)	Void	registriert ein Datenbank-Objekt
revokeObject(String)	Void	löscht ein Datenbank-Objekt aus der Liste der registrierten Datenbanken

Die einzelnen Datenquellen basieren auf dem Service `com.sun.star.sdb.DataSource` und lassen sich aus dem Datenbankkontext über die Methode `getByName` ermitteln:

```
Dim objDatabaseContext As Object
Dim objDataSource As Object
objDatabaseContext = _
   createUnoService("com.sun.star.sdb.DatabaseContext")
If objDatabaseContext.hasByName("DB_Test1") Then
   objDataSource = objDatabaseContext.getByName("DB_Test1")
   MsgBox "Zugriff erfolgt"
```

Tabelle 8.138 Die wichtigsten Eigenschaften des DataSource-Objektes:

Eigenschaft	Wert	Beschreibung
Info	Array	Array mit PropertyValue-Paaren und Verbindungsparametern (im Regelfall zumindest Benutzername und Password).
IsPasswordRequired	Boolean	das Passwort wird ständig benötigt und interaktiv vom Benutzer erfragt.
IsReadOnly	Boolean	ermöglicht ausschließlich lesende Zugriffe auf die Datenbank
Name	String	Name der Datenquelle
NumberFormatsSupplier	Object	Objekt mit den für die Datenbank verfügbaren Nummernformaten (unterstützt die Schnittstelle com.sun.star.util.XNumberFormatsSupplier, siehe Kapitel 7, Abschnitt Zahlen-, Datums- und Textformat).
Password	String	Passwort der Datenbank
SuppressVersionColumns	Boolean	unterdrückt die Anzeige von Spalten, die zur Versionsverwaltung dienen
TableFilter	Array	Liste der anzuzeigenden Tabellennamen
TableTypeFilter	Array	Liste der anzuzeigenden Tabellentypen. Möglich sind die Werte TABLE, VIEW und SYSTEM TABLE
URL	String	URL der Datenquelle in der Form jdbc: subprotocol : subname oder sdbc:subprotocol : subname
User	String	der Benutzer
Bookmarks	Object	die Sammlung der Bookmarks
QueryDefinitions	Object	die Abfragen (Queries)
Tables	Object	die Tabellen

8.5 Tabellen und Abfragen (Queries, Views)

Um auf alle Tabellen zuzugreifen und sie namentlich anzuzeigen, kann die Sammlung `Tables` verwendet werden:

```
objTabellen = objDataSource.Tables()
For i = 0 To objTabellen.Count() - 1
   objTabelle = objTabellen(i)
   strListe = strListe & Chr(13) & objTabelle.Name
Next i
MsgBox strListe
```

Analog funktioniert der Zugriff auf Abfragen über die Sammlung `QueryDefinitions`:

```
objAbfragen = objDataSource.QueryDefinitions()
For i = 0 To objAbfragen.Count() - 1
   objAbfrage = objAbfragen(i)
   strListe = strListe & Chr(13) & Chr(13) & objAbfrage.Name & _
      ":" & Chr(13) & objAbfrage.Command
Next i
MsgBox strListe
```

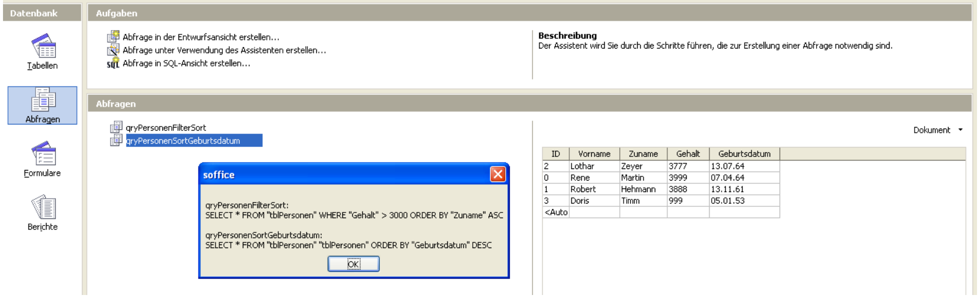


Abbildung 8.162 Alle Tabellen und Abfragen können ausgelesen werden.

Die Eigenschaft `Command` liest den SQL-Befehl der Abfragen aus.

Die Anzahl der Tabellen und Abfragen kann mit der Eigenschaft `Count` oder der Methode `getCount()` ermittelt werden.

Tabelle 8.139 Tabellen und Abfragen stellen folgende Methoden (Auswahl) zur Verfügung:

Methode	Wert	Beschreibung
hasElements(Void)	Boolean	überprüft, ob Elemente vorhanden sind
getCount(Void)	Long	die Anzahl der Elemente
getByIndex(Long)	Empty	greift auf ein Element über den Index zu
getByName(String)	Empty	greift auf ein Element über den Namen zu
getElementNames(Void)	Array	liest alle Elementnamen aus
hasByName(String)	Boolan	überprüft, ob ein bestimmtes Element vorhanden ist
replaceByName(String, Variant)	Void	benennt Tabelle oder Abfrage um
insertByName(String, Variant)	Void	fügt Tabelle oder Abfrage ein
removeByName(String)	Void	löscht Tabelle oder Abfrage
createEnumeration(Void)	Object	erstellt Aufzählungsliste

Tabelle 8.140 Abfragen verfügen über folgende wichtige Eigenschaften:

Eigenschaft	Wert	Beschreibung
Name	String	Name der Abfrage
Command	String	Das Kommando der Abfrage (SQL-Ausdruck)
EscapeProcessing	Boolean	True, wenn ein Escaping auf das Kommando angewendet werden soll.
UpdateTableName	String	Name der Tabelle, die bei einem Update verwendet werden soll.
UpdateCatalogName	String	Der Name des Tabellen-Katalogs für ein Update
UpdateSchemaName	String	Name des Tabellen-Schemas für den Update

Eine Tabelle oder eine Abfrage besitzen die Eigenschaft `Name` und die Methode `getByName(String)`, über die der Name ausgelesen werden kann.

8.6 Die Spalten

Das Objekt `Columns` liefert die Spalten einer Tabelle oder Abfrage.

```
For i = 0 To objTabelle.Columns.Count() - 1
   objSpalte = objTabelle.Columns(i)
   strListe = strListe & Chr(13) & objSpalte.Name
Next i
```

Die Eigenschaft `Count` und die Methode `getCount()` liefern leider nicht immer die korrekte Anzahl der Spalten!

8.7 Der Inhalt der Tabelle

Um nun die Daten einer Tabelle auszulesen, wird normalerweise mit dem ResultSet-Objekt gearbeitet. Es ist eine Art Zeiger, das auf einen Datensatz innerhalb einer Tabelle oder Abfrage verweist. Das folgende Beispiel zeigt, wie eine Verbindung zu einer Datenbank hergestellt wird.

```
Sub AlleDaten
   Dim objDatabaseContext As Object
   Dim objDataSource As Object
   Dim objConnection As Object
   Dim objInteractionHandler as Object
   Dim objStatement As Object
   Dim objResultSet As Object

   objDatabaseContext = _
      createUnoService("com.sun.star.sdb.DatabaseContext")
   objDataSource = objDatabaseContext.getByName("TestDB1")
   If Not objDataSource.IsPasswordRequired Then
      objConnection = objDataSource.GetConnection("","")
   Else
      objInteractionHandler = _
         createUnoService("com.sun.star.sdb.InteractionHandler")
```

```
      objConnection = _
         objDataSource.ConnectWithCompletion(objInteractionHandler)
   End If
   objStatement = objConnection.createStatement()
   objResultSet = _
      objStatement.executeQuery("SELECT ""Zuname"" FROM ""tblPersonen""")
   If Not IsNull(objResultSet) Then
      While objResultSet.next
         strListe = strListe & Chr(13) & objResultSet.getString(1)
      Wend
   End If
   MsgBox strListe
   objResultSet.Close()
   objStatement.Close()
   objConnection.Close()
End Sub
```

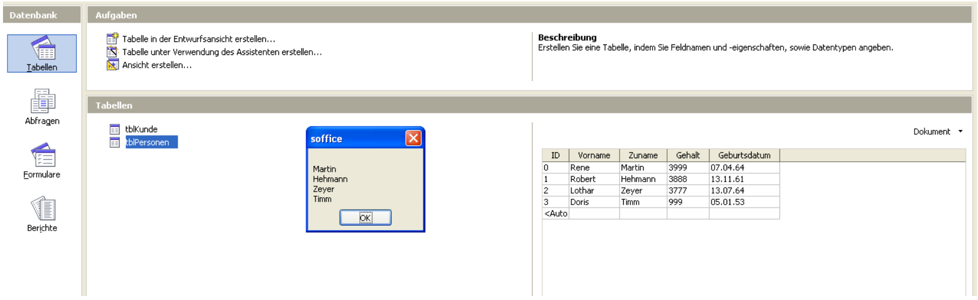


Abbildung 8.163 Die Daten können mit einem SELECT-Befehl ausgelesen werden.

Beachten Sie, dass Sie den Namen der Tabelle und der Spalte innerhalb der SQL-Anweisung in Anführungszeichen schreiben müssen. Folgende Variante führt zu einem Fehler:

```
objResultSet = _
   objStatement.executeQuery("SELECT Zuname FROM tblPersonen")
```

```
objResultSet = objStatement.executeQuery("SELECT * FROM tblPersonen")
```

Folgende Variante arbeitet korrekt:

```
objResultSet = objStatement.executeQuery("SELECT * FROM 'tblPersonen'")
```

Hinter „tblPersonen“ befindet sich zuerst ein einfaches Hochkomma (') und anschließend ein Anführungszeichen (").

Es wird eine Verbindung hergestellt, auf die das Connection-Objekt verweist. Selbstverständlich verfügt das Connection-Objekt über die Sammlung Tables, Views und Queries, mit denen die Tabellen, Sichten und Abfragen ermittelt werden können. Danach sollten Sie die Verbindung, das Statement und das ResultSet wieder sauber schließen.

Tabelle 8.141 Wichtige Methoden des Connection-Objektes:

Methode	Wert	Beschreibung
close(void)	Void	schließt die Verbindung
isClosed(void)	Boolean	prüft, ob die Verbindung geschlossen wurde
createStatement(void)	Object	generiert ein SQL-Statement
prepareStatement(String)	Object	generiert ein vorkompiliertes SQL-Statement für Parameter-Abfragen, bspw.: pstmt = con.prepareStatement("UPDATE EMPLOYEES SET SALARY = ? WHERE ID = ?") pstmt.setDouble(1, 153833.00) pstmt.setLong(2, 110592)
prepareCall(String)	Object	generiert ein „callable“ Statement
nativeSQL(String)	String	verwandelt eine SQL-Anweisung in eine interne SQL-Anweisung
setAutoCommit(Boolean)	Void	setzt den AutoCommit-Modus
getAutoCommit(void)	Boolean	liefert den AutoCommit-Modus
commit(void)	Void	führt aus
rollback(void)	Void	macht alle Änderungen rückgängig
getMetaData(void)	Object	liefert die MetaData
setReadOnly(Boolean)	Void	setzt in einen Nur-Lesen-Modus
isReadOnly(void)	Boolean	überprüft den Nur-Lesen-Status
setCatalog(String)	Void	setzt einen Catalog-Name
getCatalog(void)	String	liefert den Catalog-Namen
setTransactionIsolation(Long)	Void	setzt die Transaktion auf die Nummer
getTransactionIsolation(void)	Long	liefert die Nummer der Transaktion
getTypeMap(void)	Object	liefert das Type Map

Methode	Wert	Beschreibung
setTypeMap(Object)	Void	setzt das Type Map
getTables(void)	Object	die Tabellen
getViews(void)	Object	die Views
getQueries(void)	Object	die Abfragen (Queries)

Um einen SQL-Befehl absetzen zu können, muss ein Statement-Objekt aufgerufen werden. Es verfügt über eine Reihe von Methoden und Eigenschaften. Mit der Methode `executeQuery` kann ein SQL-Befehl ausgeführt werden. Dieses Ergebnis kann in einem Objekt gespeichert werden, das nun durchlaufen wird. Die Methode `getString` ermittelt den Wert eines Feldes – die Zahl 1 greift dabei auf die erste Spalte zurück.

Der Objekt „ResultSet“ verfügt ebenso über das Objekt „Columns“, das die korrekte Anzahl der Spalten liefert.

Beachten Sie, dass `getString` – anders als bei allen Aufzählungen in OOo – nicht bei 0, sondern bei 1 beginnt. Wird der folgende SQL-Befehl erzeugt:

```
objResultSet = objStatement.executeQuery _
   ("SELECT ""Zuname"", ""Vorname"" FROM ""tblPersonen""")
```

dann liefert

```
strListe = strListe & Chr(13) & objResultSet.getString(1)
```

die Liste der Zunamen

```
strListe = strListe & Chr(13) & objResultSet.getString(2)
```

die Liste der Vornamen und

```
strListe = strListe & Chr(13) & objResultSet.getString(0)
```

beziehungsweise

```
strListe = strListe & Chr(13) & objResultSet.getString(3)
```

erzeugen einen Fehler.

Wenn Sie eine Aktionsabfrage erzeugen möchten, dann mit der Methode `executeUpdate`:

```
objStatement.executeUpdate("INSERT INTO ""tblPersonen"" " & _
"(""Vorname"", ""Zuname"", ""Gehalt"") VALUES " & _
```

```
"('Walter', 'Poppen', 2929)")
```

Beachten Sie, dass die Werte in Anführungszeichen übergeben werden müssen. Da sie sich innerhalb einer Zeichenkette befinden, werden sie entweder in doppelte Anführungszeichen gesetzt oder in einfache Hochkommata. Der Betrag für die Zahl kann mit einem Apostroph versehen werden, muss aber nicht. Datumsangaben werden geschrieben in der Form: '1961-12-04', Dezimalzahl mit Punkt als Kommazeichen: 2.99.

So können sämtliche anderen Aktionsabfragen abgesetzt werden:

```
objStatement.executeUpdate("DROP TABLE ""Kunde""")

objStatement.executeUpdate("ALTER TABLE ""tblPersonen"" DROP ""Gehalt""")

objStatement.executeUpdate("DELETE FROM 'tblPersonen' WHERE " & _

   """Zuname"" = 'Poppen'")
```

8.7.1 Typspezifische Methoden zum Auslesen von Werten

Tabelle 8.142 Neben getString stehen noch weitere Methoden zum Zugriff auf Spalten zur Verfügung

Methode	**Rückgabewert**
getString(Long)	String
getBoolean(Long)	Boolean
getByte(Long)	Integer
getShort(Long)	Integer
getInt(Long)	Long
getLong(Long)	Integer
getFloat(Long)	Single
getDouble(Long)	Double
getBytes(Long)	Array (binäre Werte)
getDate(Long)	Object
getTime(Long)	Object
getTimestamp(Long)	Object
getBinaryStream(Long)	Object (binäre Werte)
getCharacterStream(Long)	Object
getObject(Long, Object)	Empty
getRef(Long)	Object
getBlob(Long)	Object
getClob(Long)	Object

Methode	Rückgabewert
getArray(Long)	Object

Tabelle 8.143 Zum Navigieren stehen folgende Methoden zur Verfügung:

Methode	Rückgabewert	Beschreibung
next()	Boolean	gehe zum nächsten Datensatz
previous()	Boolean	gehe zum vorhergehenden Datensatz
first()	Boolean	gehe zum ersten Datensatz
last()	Boolean	gehe zum letzten Datensatz
beforeFirst()	Void	gehe vor den ersten Datensatz
afterLast()	Void	gehe hinter den letzten Datensatz
isFirst()	Boolean	ist dieser Datensatz der erste
isLast()	Boolean	ist dieser Datensatz der letzte
isBeforeFirst()	Boolean	ResultSet steht vor dem ersten Datensatz
isAfterLast()	Boolean	ResultSet steht vor dem letzten Datensatz
getRow()	Long	greife auf den Datensatz zu
absolute(Long)	Boolean	absolute Position
relative(Long)	Boolean	relative Position

Wie aus der Liste deutlich wird, ist eine Navigation in beide Richtungen möglich, also vom Anfang zum Ende und auch vom Ende zum Anfang.

8.7.2 Ändern von Datensätzen

Wurde das ResultSet auf `UPDATABLE` gesetzt:

```
objStatement.ResultSetConcurrency    = _
   com.sun.star.sdbc.ResultSetConcurrency.UPDATABLE
```

dann können Werte in den Zellen geändert werden (die Alternative lautet: READ_ONLY). Hierfür stehen die Update-Methoden zur Verfügung, die analog zu den get-Methoden arbeiten:

Tabelle 8.144 Die Update-Methoden (jeweils ohne Rückgabewert):

Methode
updateNull(Long)
updateBoolean(Long, Boolean)
updateByte(Long, Integer)
updateShort(Long, Integer)
updateInt(Long)
updateLong(Long, Int64)
updateFloat(Long, Single)
updateDouble(Long, Double)
updateString(Long, String)
updateBytes(Long, Array)
updateDate(Long, Object)
updateTime(Long, Object)
updateTimestamp(Long, Object)
updateBinaryStream(Long, Object, Long)
updateCharacterStream(Long, Object, Long)
updateObject(Long, Variant)
updateNumericObject(Long, Variant, Long)

Wurde ein Wert geändert, dann hält der ResultSet diesen Wert. Ein Ändern in der Tabelle kann mit der Methode `cancelRowUpdates()` verhindert werden. Wichtiger ist jedoch die Methode `updateRow()`, mit der die Daten in die Tabelle geschrieben werden.

Das ResultSet stellt weiterhin verschiedene Navigationsmöglichkeiten zur Verfügung:

Tabelle 8.145 Die Konstanten vom `com.sun.star.sdbc.ResultSetConcurrency` lauten:

Konstante	Wert	Beschreibung
UPDATABLE	1008	erlaubt Änderungen
READ_ONLY	1007	erlaubt nur Lesezugriff

Tabelle 8.146 Die Konstanten vom `com.sun.star.sdbc.ResultSetType` lauten:

FORWARD_ONLY	1003	erlaubt nur eine Navigation nach vorne
SCROLL_INSENSITIVE	1004	erlaubt beliebige Navigation, keine Änderungen im Resultset möglich
SCROLL_SENSITIVE	1005	erlaubt beliebige Navigation, Änderungen im Resultset sind

		möglich

8.8 Tabellen modifizieren

8.8.1 Neue Tabellen und Abfragen erstellen

Mit dem obigen Wissen kann leicht eine neue Abfrage erstellt werden:

```
Sub Abfrage_Erstellen
   Dim objDatabaseContext As Object
   Dim objDataSource As Object
   Dim objQueryDefinitions As Object
   Dim objQueryDefinition As Object

   objDatabaseContext = _
      createUnoService("com.sun.star.sdb.DatabaseContext")
   objDataSource = objDatabaseContext.getByName("TestDB1")
   objQueryDefinitions = objDataSource.getQueryDefinitions()
   If objQueryDefinitions.hasByName("qryPersonenFilterSort") = False Then
      objQueryDefinition = _
         createUnoService("com.sun.star.sdb.QueryDefinition")
      objQueryDefinition.Command = "SELECT * FROM ""tblPersonen"" " & _
         "WHERE ""Gehalt"" > 3000 ORDER BY ""Zuname"" ASC"
      objQueryDefinitions.insertByName("qryPersonenFilterSort", " & _
         "objQueryDefinition)
   Else
      MsgBox "Die Abfrage ""qryPersonenFilterSort"" existiert bereits."
   End If
End Sub
```

Oder analog eine neue Tabelle:

```
Sub Tabelle_Erstellen
   Dim objDatabaseContext As Object
   Dim objDataSource As Object
   Dim objVerbindung as Object
   Dim objTabellenBeschreibung as Object
   Dim objSpaltenBeschreibung as Object

   objDatabaseContext = CreateUnoService _
      ( "com.sun.star.sdb.DatabaseContext" )
   objDataSource = objDatabaseContext.getByName("TestDB1")

   objVerbindung = objDataSource.getConnection("" , "")

   If objVerbindung.getTables().hasByName("tblKunde") = False Then

      objTabellenBeschreibung = objVerbindung._
         getTables().createDataDescriptor()
      objTabellenBeschreibung.Name = "tblKunde"
      objSpaltenBeschreibung = objTabellenBeschreibung._
         getColumns().createDataDescriptor()
      objSpaltenBeschreibung.Name = "ID"
      objSpaltenBeschreibung.Type = _
         com.sun.star.sdbc.DataType.INTEGER
      objSpaltenBeschreibung.IsNullable = _
         com.sun.star.sdbc.ColumnValue.NO_NULLS
      objSpaltenBeschreibung.IsAutoIncrement = True
      objTabellenBeschreibung.getColumns()._
         appendByDescriptor(objSpaltenBeschreibung)

      objSpaltenBeschreibung = objTabellenBeschreibung._
         getColumns().createDataDescriptor()
```

```
      objSpaltenBeschreibung.Name = "Auftragsstunden"
      objSpaltenBeschreibung.Precision = 12
      objSpaltenBeschreibung.Scale = 2
      objSpaltenBeschreibung.Type = _
         com.sun.star.sdbc.DataType.DECIMAL
      objSpaltenBeschreibung.IsNullable = _
         com.sun.star.sdbc.ColumnValue.NULLABLE
      objSpaltenBeschreibung.DefaultValue = 0
      objTabellenBeschreibung.getColumns()._
         appendByDescriptor(objSpaltenBeschreibung)

      objSpaltenBeschreibung = objTabellenBeschreibung._
         getColumns().createDataDescriptor()
      objSpaltenBeschreibung.Name = "Kundenname"
      objSpaltenBeschreibung.Type = _
         com.sun.star.sdbc.DataType.VARCHAR
      objSpaltenBeschreibung.Precision = 50
      objSpaltenBeschreibung.IsNullable = TRUE
      objTabellenBeschreibung.getColumns()._
         appendByDescriptor(objSpaltenBeschreibung)
      objVerbindung.getTables()._
         appendByDescriptor(objTabellenBeschreibung)
   Else
      MsgBox "Die Tabelle ""tblKunde"" existiert bereits."
   End If
End Sub
```

Abbildung 8.164 Eine neue Tabelle wird erzeugt.

Tabelle 8.147 Für die Tabellenspalten stellt der DataDescriptor folgende Eigenschaften zur Verfügung:

Eigenschaft	**Wert**	**Beschreibung**
Align	Long/void	Ausrichtung
DefaultValue	String	Vorgabewert
Description	String	Beschreibungstext
FormatKey	Long/void	Formatierung
HelpText	String/void	Hilfetext
Hidden	Boolean	verborgen
IsAutoIncrement	Boolean	die Spalte wird automatisch „hochgezählt“
IsCurrency	Boolean	Währung
IsNullable	Long	lässt Nullwerte zu. Konstante gemäß com.sun.star.sdbc.ColumnValue: NO_NULLS (0), NULLABLE (1) und NULLABLE_UNKNOWN (2)
IsRowVersion	Boolean	benutzt Zeit- oder Datumsstempel für Updates
Name	String	Name der Spalte
Precision	Long	die Anzahl der Nachkommastellen oder Zeichen des Textes
RelativePosition	Long/void	Position der Spalte
Scale	Long	die Anzahl der angezeigten Dezimalstellen
Type	Long	Der Datentyp. Konstante gemäß com.sun.star.sdbc.DataType: BIT, TINYINT, SMALLINT, INTEGER, BIGINT, FLOAT, REAL, DOUBLE, NUMERIC, DECIMAL, CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, TIMESTAMP, BINARY, VARBINARY, LONGVARBINARY, SQLNULL, OTHER, OBJECT, DISTINCT, STRUCT, ARRAY, BLOB, CLOB, REF
TypeName	String	der Name des Types
Width	Long/void	Breite der Spalte

Eigenschaft	Wert	Beschreibung

8.8.2 Einen neuen Index hinzufügen

In einer bestehenden beziehungsweise neu erstellten Tabelle kann nun ein Index erstellt werden. Damit kann ein Primärschlüssel auf ein Feld gesetzt werden:

```
objTabelle = objVerbindung.getTables().getByName("tblKunde")
objSpalte = objTabelle.Columns.getByName("ID")
objIndex = objTabelle.getIndexes().createDataDescriptor()
objIndex.Catalog = objTabelle.CatalogName
objIndex.IsUnique = True
objIndex.IsPrimaryKeyIndex = True
objIndex.Name = "ID"
objIndex.Columns.appendByDescriptor( objSpalte )
objTabelle.getIndexes().appendByDescriptor( objIndex )
```

Die Anzahl der vorhandenen Indexe kann mit `Count` oder `getCount()` gezählt werden, mit `hasByName(String)` wird die Existenz eines vorhandenen Index überprüft, mit `getByName(String)` und `getByIndex(Long)` kann auf ihn zugegriffen werden. `DropByName(String)` und `DropByIndex(Long)` löschen sie.

Tabelle 8.148 Der Index hat folgende wichtige Eigenschaften:

Eigenschaft	Wert	Beschreibung
Catalog	String	der Name des Index-Katalogs; kann leer sein
IsClustered	Boolean	
IsPrimaryKeyIndex	Boolean	Primärschlüssel
IsUnique	Boolean	eindeutig
Name	String	der Name des Index
Columns	Object	die Spalte oder Spalten, auf die er angewandt wird

Während der Index hinzugefügt wird, wird leider kein Primärschlüssel erstellt!

Datenbankzugriff

In diesem kleinen Beispiel steht die Dateneingabe und -ausgabe im Zentrum. Es wird gezeigt, wie über Dialoge bequem Daten eingegeben werden können. Diese werden in ein Tabellenblatt geschrieben. Über einen anderen Dialog erfolgt eine Suche, so dass die Daten im Nachhinein modifiziert werden können. Über einen weiteren Dialog ist eine benutzergesteuerte Auswertung der Daten möglich.

9 Beispiel 1: Ein Reporting-Tool

9.1 Die Einstellungen für den Benutzer

In einer Calc-Datei befinden sich mehrere Tabellenblätter. Nach Fertigstellung des Programms werden sie – bis auf eines – ausgeblendet:

```
For i = 0 To ThisComponent.Sheets.getCount() - 1
   If ThisComponent.Sheets.getByIndex(i).Name <> "ReportingTool" Then
      ThisComponent.Sheets.getByIndex(i).IsVisible = False
   End If
Next
```

Auf einem nicht ausgeblendeten Tabellenblatt befinden sich die Schaltflächen, mit deren Hilfe die Dialoge geöffnet werden.

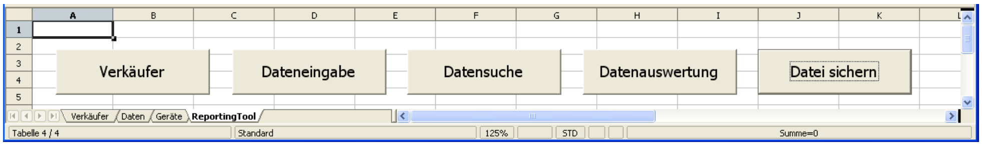


Abbildung 9.165 Die Schaltflächen und die (noch nicht ausgeblendeten) Registerblätter

9.2 Die einzelnen Dialoge

9.2.1 Informationen über die Verkäufer

Über den ersten Dialog werden neue Verkäufer eingetragen, vorhandene gelöscht oder geändert beziehungsweise Informationen zugewiesen (Zielvorgaben, Aufgabengebiet …).

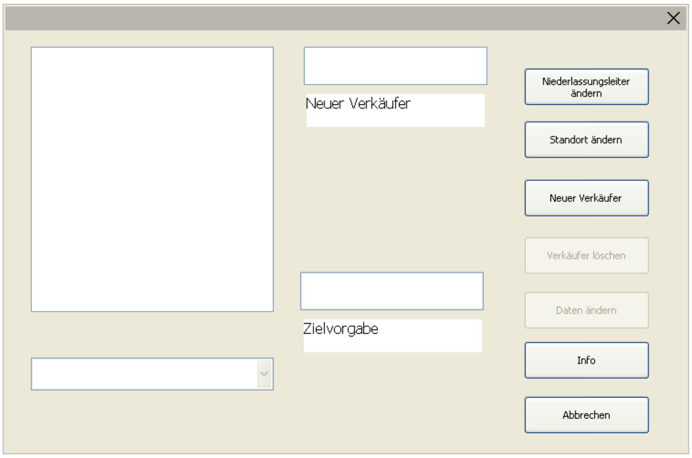


Abbildung 9.166 Der Dialog

Dabei wird der Dialog dynamisch gehalten, um dem Benutzer die Eingabe zu erleichtern:

```
Sub lstListe_Change
   With Dlg
      .getControl("cmdAendern").Label = "Daten ändern"
      .getControl("txtZielvorgabe").Enable = False
      .getControl("cboVerkaeufer").Enable = False
      .getControl("txtNeuerVerkaeufer").Visible = False
      .getControl("lblNeuerVerkaeufer").Visible = False
      .getControl("cmdNeu").Model.Label = "Neuer Verkäufer"

      If .getControl("lstListe").selectedItemPos > 0 Then
         .getControl("cmdAendern").Model.Enabled = True
         .getControl("cmdLoeschen").Model.Enabled = True
      Else
         .getControl("cmdAendern").Model.Enabled = False
         .getControl("cmdLoeschen").Model.Enabled = False
      End If
   End With
End Sub
```

Die einzelnen Eigenschaften (sichtbar und unsichtbar, aktivierbar und nicht aktivierbar (`enable`), Text anzeigen und ausblenden werden ausführlich in Kapitel 2 besprochen.

9.2.2 Informationen über die Verkaufszahlen

Über einen zweiten Dialog werden Verkaufsinformationen eingetragen. Kontrollkästchen können mit der Eigenschaft `State` abgefragt werden. Wird mit Zahlenfeldern gearbeitet, dann muss der Wert als `Value`, nicht als `String` eingetragen werden. Etwas umständlich ist das Auslesen und Eintragen von Datumsangaben aus Datumsfeldern – sie müssen mit der Funktion `CDateFromIso` konvertiert werden:

```
objBlattDaten.getCellByPosition(12, intZaehler).Value = _
   CDateFromIso(Dlg.getControl("txtBestelldatum").Model.Date)
objBlattDaten.getCellByPosition(12, intZaehler).NumberFormat = _
   lngNumberFormatId
```

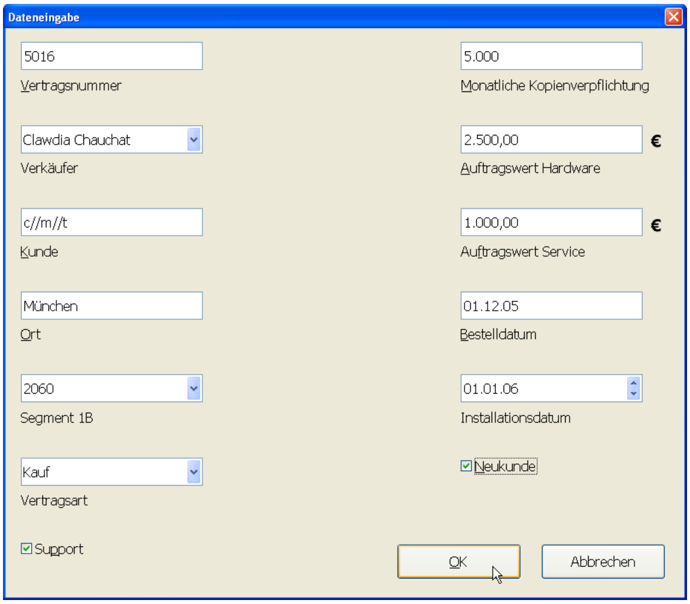


Abbildung 9.167 Die verkauften Geräte werden erfasst.

Die Datenliste der Geräte wird aus einem (verborgenen) Tabellenblatt ausgelesen und in diesem Dialog angezeigt.

```
Sub DatenEingabe
   Dim objCursorGeraete As Object
   Dim intZeilen As Integer
   Dim intZaehler As Integer

   DialogLibraries.LoadLibrary("Standard")
   Dlg = CreateUnoDialog(DialogLibraries.Standard.dlgDatenMaske
   objBlattGeraete = ThisComponent.Sheets.getByName("Geräte")
    objZelleGeraete = objBlattGeraete.getCellByPosition(0, 0)
   objCursorGeraete = _
      objBlattGeraete.createCursorByRange(objZelleGeraete)
   objCursorGeraete.collapseToCurrentRegion()
   intZeilen = objCursorGeraete.Rows.Count
   For intZaehler = 1 To intZeilen -1
      Dlg.getControl("lstGeraete").addItem _
         (objBlattGeraete.getCellByPosition(0, intZaehler).String, _
         intZaehler - 1)
   Next
   Dlg.getControl("lstGeraete").selectItemPos(0, True)
   Dlg.getControl("lblSegmente").Text = _
      objBlattGeraete.getCellByPosition(1, 1).String

   Dlg.Execute()

End Sub
```

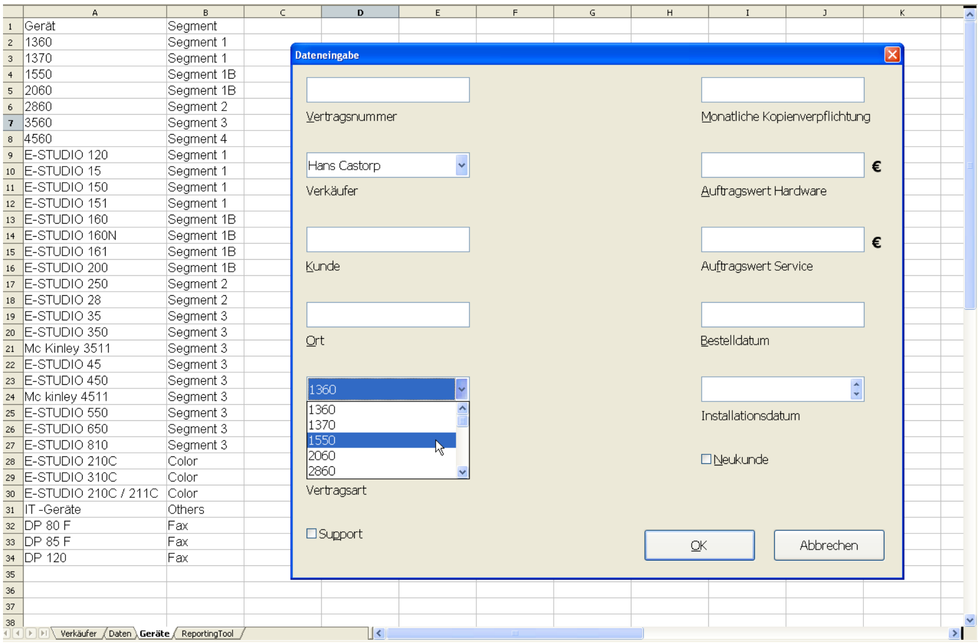


Abbildung 9.168 Die benötigten Daten werden von einem der Blätter ausgelesen.

9.2.3 Ein Dialog zur Datensuche

Für die Datensuche wird die gleiche Maske wie für die Datensuche verwendet. Im ersten Schritt werden sämtliche Steuerelemente ausgeblendet, im zweiten Schritt die benötigten wieder eingeblendet:

```
For intZaehler = 0 To UBound(Dlg.Controls)
   Dlg.Controls(intZaehler).Visible = False
Next

Dlg.getControl("cmdSuchen").Visible = True
Dlg.getControl("txtVertragsnummer").Visible = True
Dlg.getControl("lblVertragsnummer").Visible = True
[...]
Dlg.getControl("cmdSuchen").Visible = True
Dlg.getControl("cmdOK").Label = "Ändern"
```

```
Dlg.Execute()
```

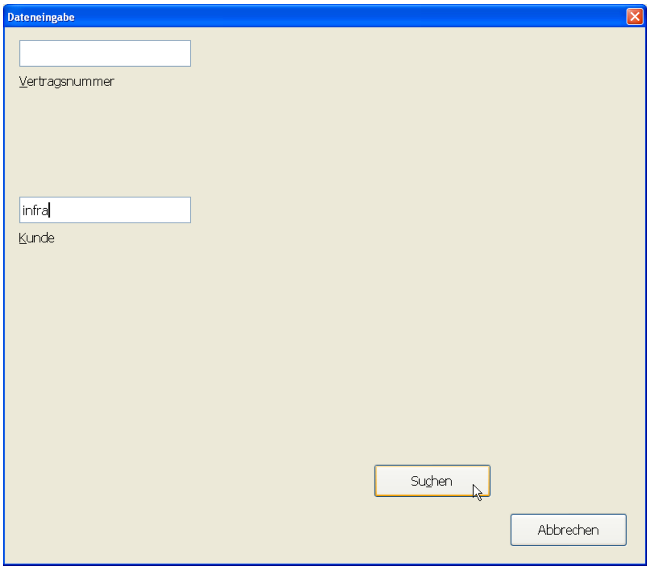


Abbildung 9.169 Datensätze können gesucht werden.

Der Benutzer gibt sein Suchkriterium ein. Werden Daten gefunden, dann werden die Daten angezeigt:

```
With objBlattDaten
   Dlg.getControl("txtVertragsnummer").Text = _
      .getCellByPosition(0,  intZaehler).String
   Dlg.getControl("lstVerkaeufer").selectItem _
      (objBlattDaten.getCellByPosition(2,  intZaehler).String, True)
   Dlg.getControl("txtKunde").Text = _
      .getCellByPosition(3,  intZaehler).String
   Dlg.getControl("txtOrt").Text = _
      .getCellByPosition(4,  intZaehler).String
   Dlg.getControl("lstGeraete").selectItem _
      (objBlattDaten.getCellByPosition(5,  intZaehler).String, True)
   Dlg.getControl("lblSegmente").Text = _
      .getCellByPosition(6,  intZaehler).String
```

```
    Dlg.getControl("lstVertragsart").selectItem _
        (objBlattDaten.getCellByPosition(7,  intZaehler).String, True)
    Dlg.getControl("chkSupport").State = _
        .getCellByPosition(8,  intZaehler).Value
    Dlg.getControl("txtMonatlicheKopienverpflichtung").Value = _
        .getCellByPosition(9,  intZaehler).Value
    Dlg.getControl("txtHardware").Value = _
        .getCellByPosition(10, intZaehler).Value
    Dlg.getControl("txtService").Value = _
        .getCellByPosition(11, intZaehler).Value

    Dlg.getControl("txtBestelldatum").Model.Date = _
        CDateToIso(objBlattDaten.getCellByPosition(12, intZaehler).Value)
    Dlg.getControl("txtInstallationsdatum").Model.Date = _
        CDateToIso(objBlattDaten.getCellByPosition(13, intZaehler).Value)

    Dlg.getControl("chkNeukunde").State = _
        .getCellByPosition(14, intZaehler).Value
End With
```

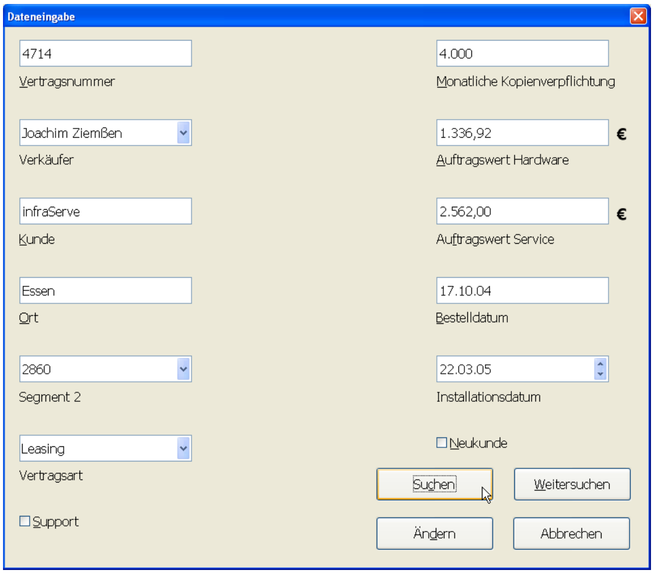


Abbildung 9.170 Der Suchen-Dialog mit gefundenen Daten (der gleiche Dialog wie „Dateneingabe“)

9.2.4 Auswertungen

Und nun stellen mögliche Auswertungen auch kein Problem mehr dar:

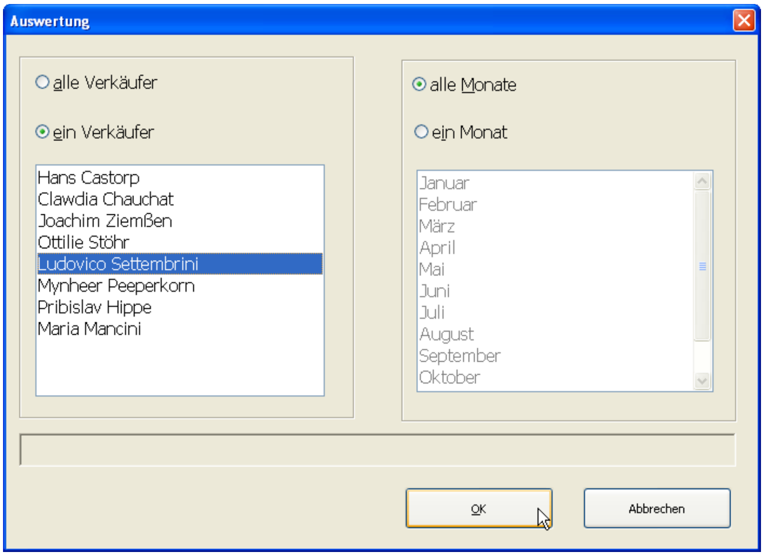


Abbildung 9.171 Der Dialog zur Datenfilterung

Wird beispielsweise ein Monat ausgewertet, dann kann „gefiltert“ werden:

```
If Month(objBlattDaten.getCellByPosition(12, intZaehlerZeilen).Value) = _
   intListenPos + 1 Then
```

Zellen können beschriftet und formatiert werden:

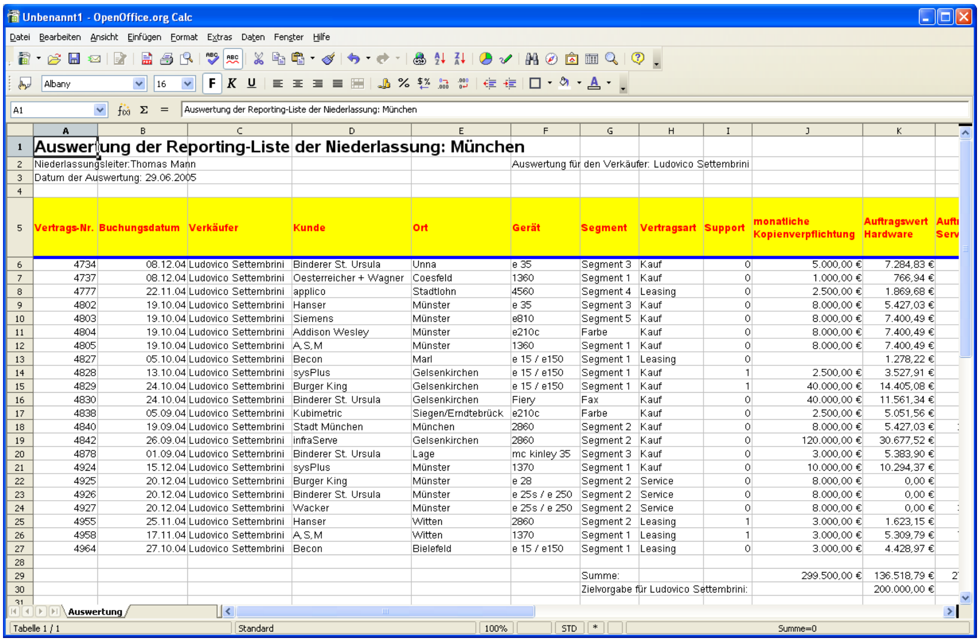
```
objLinie.Color = RGB(0, 0, 255)
objLinie.OuterLineWidth = 100
With objBlattNeu.getCellByPosition(intZaehlerSpalten, _
   intZaehlerZeilenNeu + 4)
   .CharWeight = com.sun.star.awt.FontWeight.BOLD
   .CharColor = RGB(255, 0, 0)
   .CellBackColor = RGB(255, 255, 0)
   .BottomBorder = objLinie
   .VertJustify = com.sun.star.table.CellVertJustify.CENTER
End With
```

Spalten können verbreitert werden:

```
For intZaehlerSpalten = 1 To 14
   objBlattNeu.Columns(intZaehlerSpalten).OptimalWidth = True
```

```
Next
```

Das Ergebnis kann sich sehen lassen:



Auswertung der Reporting-Liste der Niederlassung: München

Niederlassungsleiter:Thomas Mann

Datum der Auswertung: 29.06.2005

Auswertung für den Verkäufer: Ludovico Settembrini

Vertrags-Nr.	Buchungsdatum	Verkäufer	Kunde	Ort	Gerät	Segment	Vertragsart	Support	monatliche Kopienverpflichtung	Auftragswert Hardware	Auftr Serv
4734	08.12.04	Ludovico Settembrini	Binderer St. Ursula	Unna	e 35	Segment 3	Kauf	0	5.000,00 €	7.284,83 €	
4737	08.12.04	Ludovico Settembrini	Oesterreicher + Wagner	Coesfeld	1360	Segment 1	Kauf	0	1.000,00 €	766,94 €	
4777	22.11.04	Ludovico Settembrini	applico	Stadtlohn	4560	Segment 4	Leasing	0	2.500,00 €	1.869,68 €	
4802	19.10.04	Ludovico Settembrini	Hanser	Münster	e 35	Segment 3	Kauf	0	8.000,00 €	5.427,03 €	
4803	19.10.04	Ludovico Settembrini	Siemens	Münster	e810	Segment 5	Kauf	0	8.000,00 €	7.400,49 €	
4804	19.10.04	Ludovico Settembrini	Addison Wesley	Münster	e210c	Farbe	Kauf	0	8.000,00 €	7.400,49 €	
4805	19.10.04	Ludovico Settembrini	A,S,M	Münster	1360	Segment 1	Kauf	0	8.000,00 €	7.400,49 €	
4827	05.10.04	Ludovico Settembrini	Becon	Marl	e 15 / e150	Segment 1	Leasing	0		1.278,22 €	
4828	13.10.04	Ludovico Settembrini	sysPlus	Gelsenkirchen	e 15 / e150	Segment 1	Kauf	1	2.500,00 €	3.527,91 €	
4829	24.10.04	Ludovico Settembrini	Burger King	Gelsenkirchen	e 15 / e150	Segment 1	Kauf	1	40.000,00 €	14.405,08 €	
4830	24.10.04	Ludovico Settembrini	Binderer St. Ursula	Gelsenkirchen	Fiery	Fax	Kauf	0	40.000,00 €	11.561,34 €	
4838	05.09.04	Ludovico Settembrini	Kubimetric	Siegen/Erndtebrück	e210c	Farbe	Kauf	0	2.500,00 €	5.051,56 €	
4840	19.09.04	Ludovico Settembrini	Stadt München	München	2860	Segment 2	Kauf	0	8.000,00 €	5.427,03 €	
4842	26.09.04	Ludovico Settembrini	infraServe	Gelsenkirchen	2860	Segment 2	Kauf	0	120.000,00 €	30.677,52 €	
4878	01.09.04	Ludovico Settembrini	Binderer St. Ursula	Lage	mc kinley 35	Segment 3	Kauf	0	3.000,00 €	5.383,90 €	
4924	15.12.04	Ludovico Settembrini	sysPlus	Münster	1370	Segment 1	Kauf	0	10.000,00 €	10.294,37 €	
4925	20.12.04	Ludovico Settembrini	Burger King	Münster	e 28	Segment 2	Service	0	8.000,00 €	0,00 €	
4926	20.12.04	Ludovico Settembrini	Binderer St. Ursula	Münster	e 25s / e 250	Segment 2	Service	0	8.000,00 €	0,00 €	
4927	20.12.04	Ludovico Settembrini	Wacker	Münster	e 25s / e 250	Segment 2	Service	0	8.000,00 €	0,00 €	
4955	25.11.04	Ludovico Settembrini	Hanser	Witten	2860	Segment 2	Leasing	1	3.000,00 €	1.623,15 €	
4958	17.11.04	Ludovico Settembrini	A,S,M	Witten	1370	Segment 1	Leasing	1	3.000,00 €	5.309,79 €	
4964	27.10.04	Ludovico Settembrini	Becon	Bielefeld	e 15 / e150	Segment 1	Leasing	0	3.000,00 €	4.428,97 €	
						Summe:			299.500,00 €	136.518,79 €	2
						Zielvorgabe für Ludovico Settembrini:				200.000,00 €	

Abbildung 9.172 Die ausgewerteten Daten

9.2.5 Daten sichern

Damit der Benutzer seine Dateien sichern kann oder die Reporting-Datei für das nächste Jahr als „leere“ Datei an einen anderen Ort gespeichert werden kann, stellt OOo-Basic die Services „com.sun.star.ui.dialogs.FilePicker“ und „com.sun.star.ui.dialogs.FolderPicker“ zur Verfügung. Damit kann ein Dialog angezeigt werden, über den der Benutzer einen Ordner auswählt:

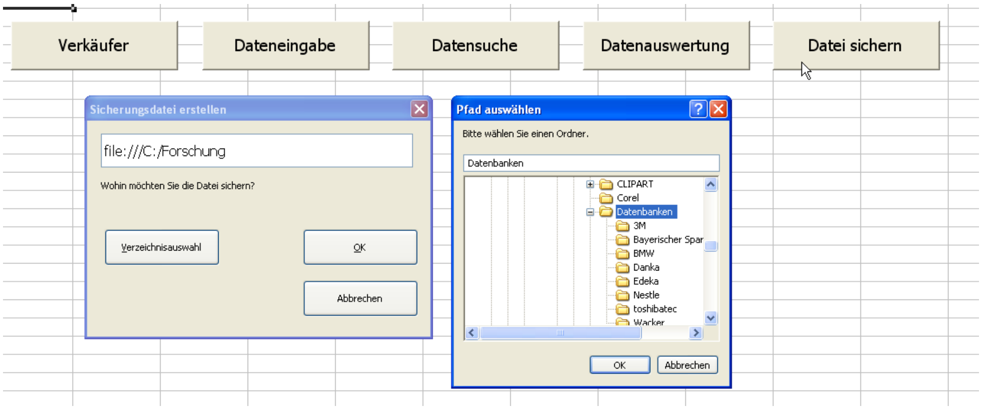


Abbildung 9.173 Die „Datenbank" kann einfach gesichert werden.

```
objVerzeichnisDlg = _
   CreateUnoService("com.sun.star.ui.dialogs.FolderPicker")
If objVerzeichnisDlg.execute() Then
   Dlg.getControl("txtPfad").Text = objVerzeichnisDlg.Directory
End If
```

9.2.6 Fazit

In diesem Beispiel wurde gezeigt, wie eine Dateneingabe und -ausgabe bequem über Dialoge gesteuert werden kann. Der Benutzer bekommt die gespeicherten Daten nie zu sehen, das heißt, er kann sie auch nicht „aus Versehen" zerstören.

Sicherlich gestaltet sich die Wirklichkeit in Bereichen wie Marketing, Vertrieb, Controlling, Buchhaltung, Personalwesen ... komplexer als in diesem Beispiel. Dennoch wurden hier die wichtigsten Elemente beschrieben, mit deren Hilfe sich komplexere Beispiele erstellen lassen.

Beispiel 1: Ein Reporting-Tool

In einer großen Firma werden Tausende von Vorlagen auf dem Server abgelegt, auf dem jeder Benutzer Leserechte hat. Jeder der Anwender soll mit den Vorlagen in Writer arbeiten. Damit der einzelne Anwender nicht den Überblick verliert, wird hierzu ein „intelligenter" Formularmanager erstellt, bei dem der Anwender über die Auswahl einer bestimmten Abteilung oder eines Sachgebiets jeweils die korrekten und aktuellen Vorlagen sieht.

Dieses Beispiel wurde für Windows konzipiert – für Linux müssten ein paar Kleinigkeiten geändert werden.

10 Der Formular-Explorer

10.1 Der Aufbau

Auf einem Dialog befinden sich ein Listenfeld, in dem später die Vorlagennamen angezeigt werden, mehrere Optionsfelder, die entweder „sprechende“ Namen haben („optStandort“, „optInternational“, „optEigene“ und „optFav“) oder mit „optHx“ beziehungsweise „optUxx“ durchnummeriert sind. Am unteren Rand befinden sich Befehlsschaltflächen:

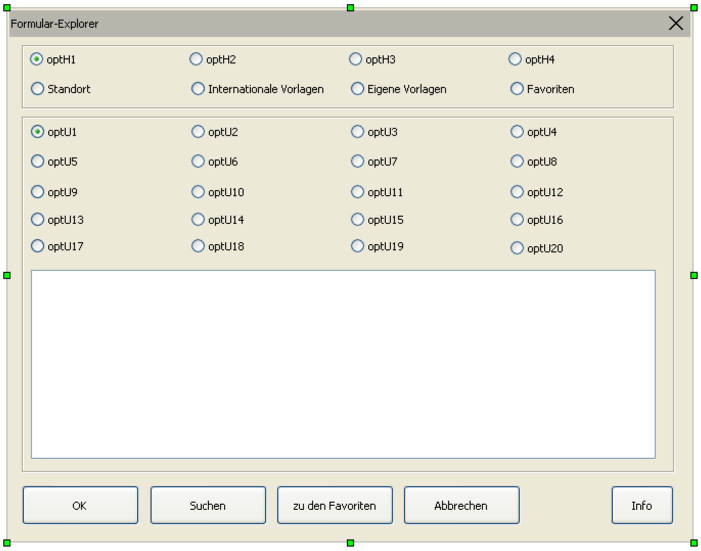


Abbildung 10.174 Der Aufbau des Dialogs

Der Dialog wird gestartet, einige Labels beschriftet:

```
DlgExplorer = CreateUnoDialog(DialogLibraries.Explorer.dlgExplorer)
DlgSuche = CreateUnoDialog(DialogLibraries.Explorer.dlgSuche)
DlgExplorer.getControl("optH1").Model.Label = "Interne Vorlagen"
DlgExplorer.getControl("optH2").Model.Label = "Externe Vorlagen"
DlgExplorer.getControl("optH3").Model.Label = "Allgemeine Vorlagen"
```

```
   DlgExplorer.getControl("optH4").Model.Label = _
      "Richtlinien und Vorgaben"
   DlgExplorer.getControl("optStandort").Model.Label = _
      StandortBestimmen()
```

Die Funktion `StandortBestimmen()` ermittelt den aktuellen Standort, der in einer Textdatei eingetragen ist:

```
Function StandortBestimmen() As String

   Dim strStandort As String
   Dim strKuerzel As String
   Dim strPfad As String
   strPfad = Environ("windir")
   If Right(strPfad, 1) <> "/" Then
      strPfad = strPfad & "/"
   End If

   strPfad = strPfad & "ExplorerStandort.txt"

   If Dir(strPfad, 0) = "" Then
      StandortBestimmen = "München"
   Else
      Open strPfad For Input As #1
            Input #1, strStandort, strKuerzel
            StandortBestimmen = strStandort

      Close #1

   End If

End Function
```

Im Anschluss werden die zugehörigen Ordner des Standorts angezeigt:

```
Sub OrdnerAnzeigen
```

```
Dim intZaehler As Integer
Dim intZaehler2 As Integer
Dim intOptButs As Integer

intZaehler = 1
strOrdner = Dir(strVorlagenPfad, 16)
DlgExplorer.getControl("cmdFav").Model.Label = "zu den Favoriten"
Do
   If strOrdner <> "." and strOrdner <> ".." Then
      If (GetAttr(strVorlagenPfad & strOrdner) AND 16) >0 then
         If Left(strOrdner, 2) = strPraefix Then

             DlgExplorer.getControl("optU" & intZaehler).Visible = _
                True
             DlgExplorer.getControl("optU" & intZaehler).Label = _
                Mid(strOrdner, 4)
             DlgExplorer.getControl("optU" & _
                intZaehler).Model.HelpText = Mid(strOrdner, 4)
              intZaehler = intZaehler + 1
              If intZaehler > 20 Then
                 Exit Do
              End If
         End If
      End If
   End If
   strOrdner = Dir

Loop Until strOrdner = ""
intOptButs = intZaehler

For intZaehler2 = intZaehler To 20
   DlgExplorer.getControl("optU" & intZaehler2).Visible = False
```

```
    Next
    DlgExplorer.getControl("optU1").State = True
    DlgExplorer.getControl("cmdOK").Model.Enabled = False
    DlgExplorer.getControl("cmdFav").Model.Enabled = False
     Call ListeGrossKlein(intOptButs)
    Call DateienAnzeigen
     DlgExplorer.Execute()

End Sub
```

Das Ergebnis sieht dann folgendermaßen aus, wenn der ausgelesene Standort „Frankfurt“ lautet:

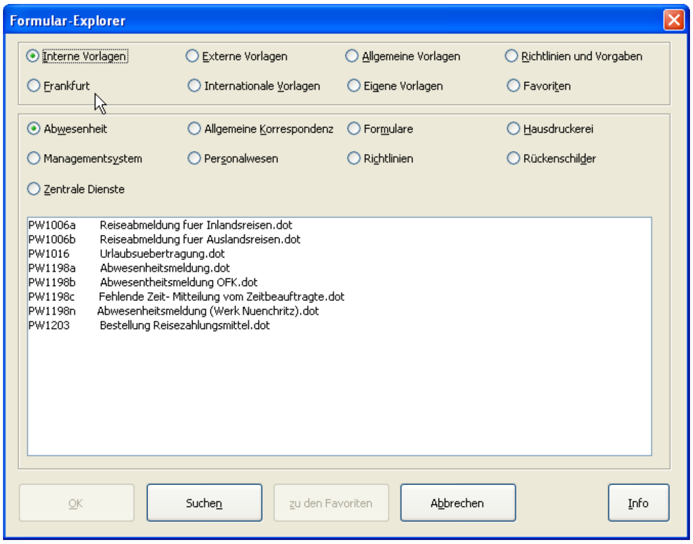


Abbildung 10.175 Bild vom gestarteten Dialog

Die ersten vier Optionsbuttons „füllen“ das Präfix mit „in“, „ex“, „al“ oder „ri“, ebenso der „internationale Button „it“:

```
Sub optH1_Click
    strPraefix = "in"
    Call OrdnerAnzeigen
End Sub
```

und so weiter.

Ein Klick auf den entsprechenden Optionsbutton „füllt“ das Listenfeld mit den Namen der entsprechenden Vorlagen:

```
Sub DateienAnzeigen
   Dim intFelderZahl As Integer
   Dim intFeldZaehler As Integer
   Dim strFeldName As String
   Dim intZaehler As Integer
   Dim intZaehler2 As Integer
   Dim strDatei As String

   intFelderZahl = UBound(DlgExplorer.Controls)

   For intFeldZaehler = 0 To intFelderZahl
      strFeldName = DlgExplorer.Model.ElementNames(intFeldZaehler)
      If Left(strFeldName, 4) = "optU" Then
         If DlgExplorer.getControl(strFeldName).State = True Then
            strOrdner = strPraefix & "_" & _
               DlgExplorer.getControl(strFeldName).Model.Label
            Exit For
         End If
      End If
   Next

   intZaehler = 1

   DlgExplorer.getControl("lstListe").removeItems _
      (0, DlgExplorer.getControl("lstListe").ItemCount)

   strDatei = Dir(strVorlagenPfad & strOrdner & "/",0)
   Do
      If strDatei <> "." and strOrdner <> ".." Then
```

```
        If (GetAttr(strVorlagenPfad & strOrdner & "/" & strDatei) _
           AND 0) = 0 then
           DlgExplorer.getControl("lstListe").addItem(strDatei, _
              intZaehler)
           intZaehler = intZaehler + 1
        End If
     End If
     strDatei = Dir

  Loop Until strDatei = ""

End Sub
```

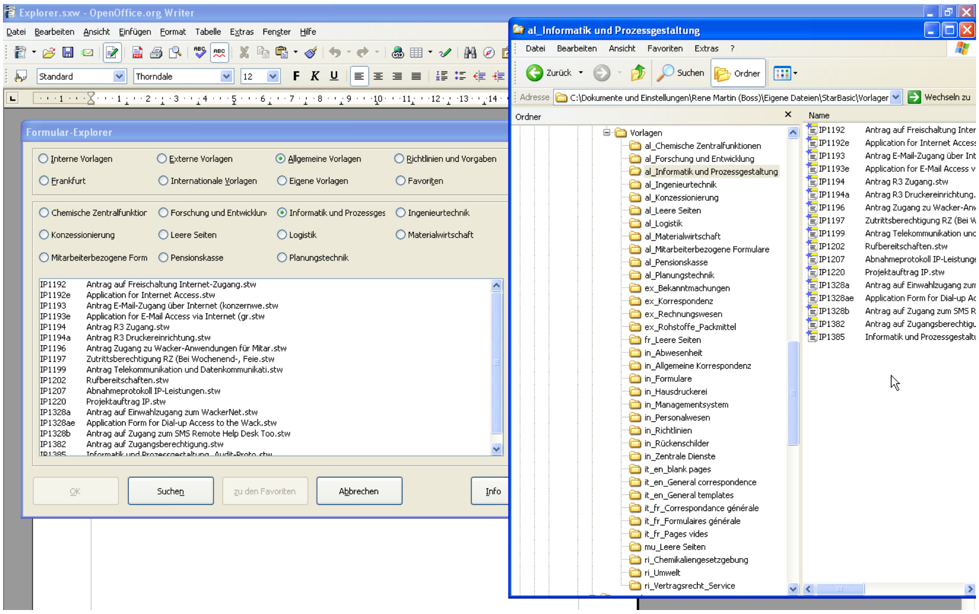


Abbildung 10.176 Die Dateinamen eines Ordners werden ausgelesen.

Damit der Platz auf dem Dialog optimal ausgenutzt wird, wird das Listenfeld dynamisch vergrößert und verkleinert – die entsprechenden Werte sind schnell durch etwas Probieren herausgefunden:

```
Sub ListeGrossKlein(intAnzahl As Integer)
   intAnzahl = intAnzahl / 4
   DlgExplorer.getControl("lstListe").Visible = False
   Select Case intAnzahl
   Case 0
      DlgExplorer.getControl("lstListe").Model.PositionY = 50
      DlgExplorer.getControl("lstListe").Model.Height = 204
   Case 1
      DlgExplorer.getControl("lstListe").Model.PositionY = 70
      DlgExplorer.getControl("lstListe").Model.Height = 184
   Case 2
      DlgExplorer.getControl("lstListe").Model.PositionY = 90
      DlgExplorer.getControl("lstListe").Model.Height = 164
   Case 3
      DlgExplorer.getControl("lstListe").Model.PositionY = 110
      DlgExplorer.getControl("lstListe").Model.Height = 144
   Case 4
      DlgExplorer.getControl("lstListe").Model.PositionY = 130
      DlgExplorer.getControl("lstListe").Model.Height = 124
   Case 5, 6
      DlgExplorer.getControl("lstListe").Model.PositionY = 150
      DlgExplorer.getControl("lstListe").Model.Height = 104
   End Select
   DlgExplorer.getControl("lstListe").Visible = True
End Sub
```

Selbstverständlich werden auch die Buttons bei Bedarf aktiv oder inaktiv (dies wird mit der Eigenschaft `Model.Enabled` realisiert).

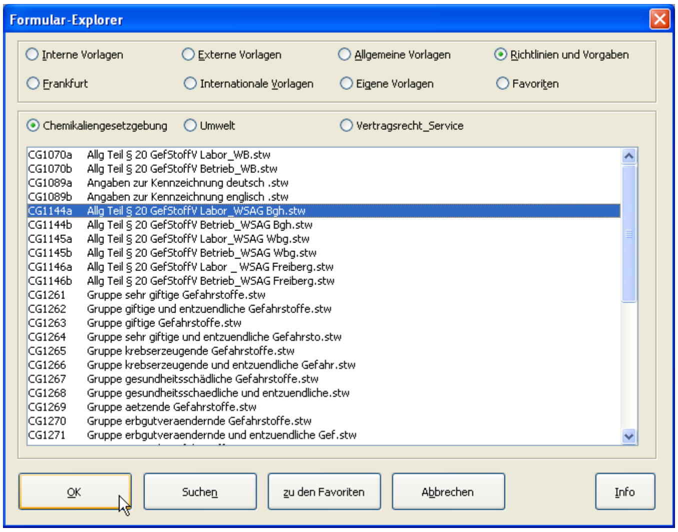


Abbildung 10.177 Der dynamische Dialog

Das Schließen ist kein Problem; der Suchen-Button führt zu einem zweiten Dialog. Dort kann der Benutzer eine Vorlage über den Dateinamen oder einen Inhalt (oder beides) suchen.

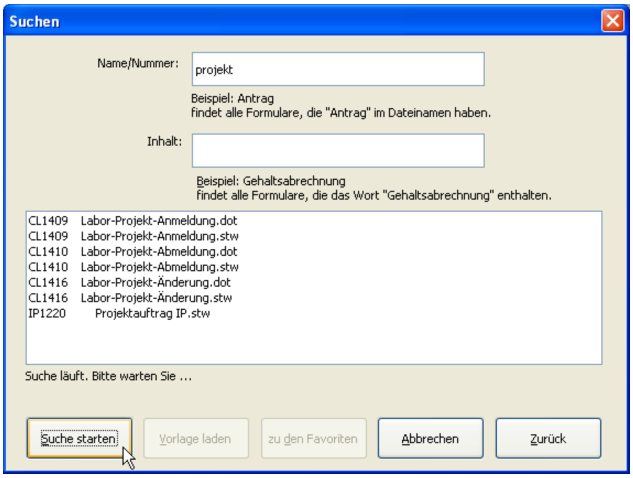


Abbildung 10.178 Der Suchen-Dialog

Der zentrale Code durchläuft die Dateinamen und sucht nach dem eingegebenen Dateinamen oder öffnet die Dateinamen und vergleicht die Inhalte:

```
Sub SucheStarten_Click
   Dim strDatei As String
   Dim intOrdnerZaehler As Integer
   Dim intZaehler As Integer

   If DlgSuche.getControl("txtName").getText() = "" _
      And DlgSuche.getControl("txtInhalt").getText() = "" Then
      MsgBox "Sie müssen schon einen Suchtext eingeben!"
      DlgSuche.getControl("txtName").setFocus()
   Else
      DlgSuche.getControl("lblInfo").Model.Label = _
         "Suche läuft. Bitte warten Sie ..."
      DlgSuche.getControl("lstListe").removeItems(0, _
         DlgSuche.getControl("lstListe").ItemCount)
      ' -- leere die Liste
      intZaehler = 0
      strOrdner = Dir(strVorlagenPfad, 16)
      Do While strOrdner <> ""
         If strOrdner <> "." and strOrdner <> ".." Then
            If (GetAttr(strVorlagenPfad & strOrdner) AND 16) > 0 then
               ReDim Preserve strAlleOrdner(intOrdnerZaehler)
                strAlleOrdner(intOrdnerZaehler) = strVorlagenPfad & _
                   strOrdner & "/"
                intOrdnerZaehler = intOrdnerZaehler + 1
            End If
         End If
         strOrdner = Dir
      Loop
      ' -- die Unterordner werden ausgelesen
      ' -- und in dem Datenfeld "strAlleOrdner" gespeichert.

      Call DateienSuche
```

```
      DlgSuche.getControl("lblInfo").Model.Label = _
      "Suche beendet. Es wurde" & IIf(intDateien = 1,"", "n") & " " & _
      intDateien & " Datei" & IIf(intDateien = 1,"", "en") & " gefunden."
      DlgSuche.getControl("cmdFav").Model.Enabled = False
      DlgSuche.getControl("cmdVorlage").Model.Enabled = False
   End If

End Sub

Sub DateienSuche
   Dim intZaehler As Integer
   Dim strDatei As String

   intDateien = 0
   For intZaehler = 0 To UBound(strAlleOrdner())
   ' -- durchlaufe alle Ordner
      strDatei = Dir(strAlleOrdner(intZaehler), 0)
      Do While strDatei <> ""
         If strDatei <> "." And strDatei <> ".." Then
         ' -- übergehe "." und ".."
            If (GetAttr(strAlleOrdner(intZaehler) & strDatei) AND 0) _
               = 0 Then
            ' -- wenn eine Datei gefunden wird
               If InStr(strDatei, _
                  DlgSuche.getControl("txtName").getText()) > 0 Then
                   If DateienSucheUeberInhalt(strAlleOrdner(intZaehler) _
                      & strDatei) = True Then
                     DlgSuche.getControl("lstListe").AddItem(strDatei, _
                        intDateien)
                     ReDim Preserve strPfadGanz(intDateien)
                     strPfadGanz(intDateien) = _
```

```
                        strAlleOrdner(intZaehler) & strDatei
                      intDateien = intDateien + 1
                   End If
               End If
            End If
         End If
         strDatei = Dir

      Loop
   Next
End Sub

Function DateienSucheUeberInhalt(strDateiName As String) As Boolean
   Dim blnGefunden As Boolean
   Dim strInhalt As String
   On Error Resume Next
   If DlgSuche.getControl("txtInhalt").getText() = "" Then
      DateienSucheUeberInhalt = True
   Else
      blnGefunden = False
      DlgSuche.getControl("lblInfo").Model.Label = "öffne: " & _
         strDateiName
       Open strDateiName For Input As #1
          Do While Not EOF(1)
            Input #1, strInhalt
            If InStr(strInhalt, _
               DlgSuche.getControl("txtInhalt").getText()) > 0 Then
               blnGefunden = True
                Exit Do
            End If
           Loop
      Close #1
```

```
        DateienSucheUeberInhalt = blnGefunden
    End If
End Function
```

Auch dieser Dialog ist dynamisch, um dem Benutzer zu „helfen“, dass er nur Dateien öffnet, die wirklich angezeigt werden.

Übrigens hat der Benutzer die Möglichkeit, seine für ihn wichtigen Vorlagen in einer Favoriten-Datei zu speichern, die er sich jederzeit anzeigen lassen oder auch einzelne Einträge löschen kann.

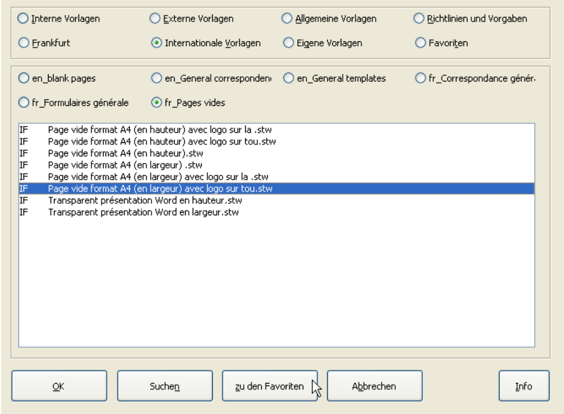


Abbildung 10.179 Eine Vorlage wird zu den Favoriten aufgenommen.

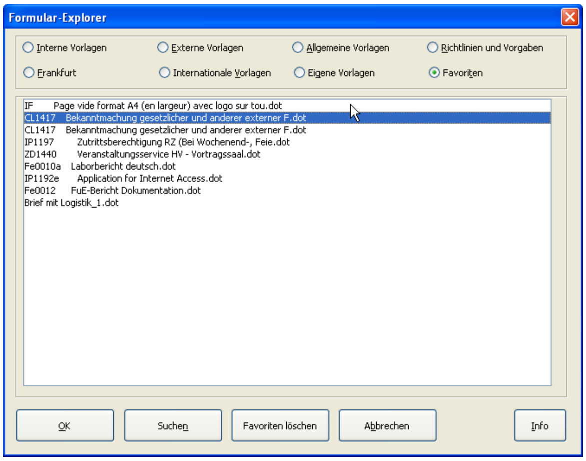


Abbildung 10.180 Die Favoriten werden angezeigt.

Und schließlich kann er persönliche Informationen über eine weitere Maske hinterlegen, die automatisiert in bestimmte Vorlagen eingetragen werden.

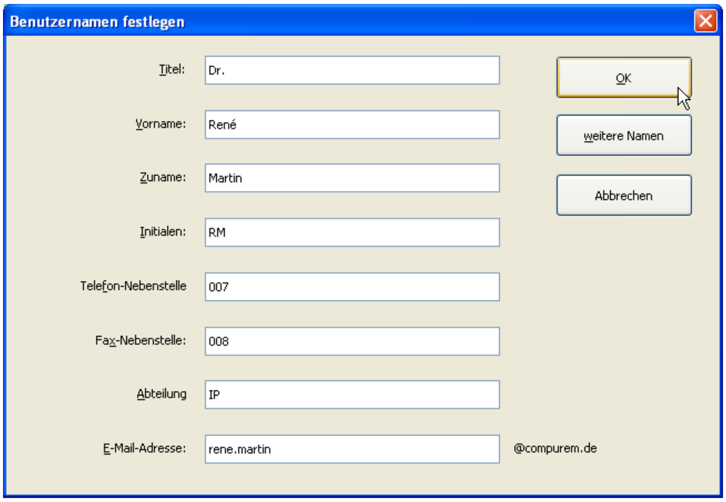


Abbildung 10.181 Der Dialog für die Benutzernamen

Dabei werden die Initialen und die Mailadresse automatisch generiert:

```
Sub MailEintragen

   Dim strNameGanz As String
   Dim strName() As String
   Dim i As Integer

   strNameGanz = DlgNamen.getControl("txtVorname").Model.Text & _
      "." & DlgNamen.getControl("txtZuname").Model.Text


   i = Len(strNameGanz)
   ReDim strName(i)
```

```
For i = 1 To Len(strNameGanz)
   strName(i) = Mid(strNameGanz, i, 1)
Next

For i = 1 To Len(strNameGanz)
   Select Case Mid(strNameGanz, i, 1)
      ' -- Kleinbuchstaben:
      Case "ä"
         strName(i) = "ae"
      Case "ö", "æ", "ø"
         strName(i) = "oe"
      Case "ü"
         strName(i) = "ue"
      Case "ß"
         strName(i) = "ss"
      Case "ç"
         strName(i) = "c"
      Case "à", "á", "â", "ã", "å"
         strName(i) = "a"
      Case "ë", "é", "è"
         strName(i) = "e"
      Case "ì", "í", "î", "ï"
         strName(i) = "i"
      Case "ñ"
         strName(i) = "n"
      Case "ò", "ó", "ô", "õ"
         strName(i) = "o"
      Case "ù", "ú", "û"
         strName(i) = "u"
```

```
         Case "ÿ", "ý"
...
   For i = 1 To Len(strNameGanz)
      strName(0) = strName(0) & LCase(strName(i))
   Next

   DlgNamen.getControl("txtMail").Model.Text = strName(0)
   DlgNamen.getControl("txtInitialen").Model.Text = _
      Left(DlgNamen.getControl("txtVorname").Model.Text, 1) & _
      Left(DlgNamen.getControl("txtZuname").Model.Text, 1)
End Sub
```

Der Formular-Explorer

Formulare spielen eine wichtige Rolle in Unternehmen. Damit nicht immer gleiche oder ähnliche Texte geschrieben werden müssen, können Texte auch generiert werden, so dass schnell eine große Anzahl an Schreiben erstellt werden kann. Dieses Kapitel stellt Ihnen mehrere praxisnahe Beispiele vor, wie solche Dateien generiert werden können.

11 Intelligente Formulare

11.1 Der Entgeltbrief I

Im ersten Beispiel werden Daten in einen Dialog eingegeben. Vor dem Erstellen des Dokumentes werden nicht nur bestimmte Anforderungen der eingegebenen Daten überprüft, sondern die Daten werden danach korrekt (beispielsweise als formatierte Zahlen) in ein Writer-Dokument eingetragen. Dort können nicht nur einfache Informationen an bestimmte Stellen geschrieben werden – manchmal muss auch eine Tabelle generiert werden, Text gelöscht werden und so weiter.

Im ersten Schritt wird der Dialog erstellt. Da er sehr viele Kontrollfelder besitzt, müssen sie entweder gut benannt werden (damit sie später beim Zugriff keine Schwierigkeiten erhalten), oder Sie denken sich einen Index aus, über den Sie mit Hilfe eines Zählers zugreifen können: „txtZulage01“, „txtZulage02“, „txtZulage03“ …

Die exakten Abstände wurden mit Hilfe der Eigenschaften realisiert – dies ist zugegebenermaßen eine Menge Arbeit! Auch wenn alle Eingabefelder mit dem Präfix „txt“ beginnen, sind „txtDatum“ und „txtStufungsdatum“ Datumsfelder; sämtliche Felder für Geldbeträge sind Währungsfelder. Dort wurde als minimaler Wert 0 eingetragen, zwei Nachkommastellen und Tausendertrennzeichen festgelegt, ebenso wie das Währungssymbol auf „€“ gesetzt wurde. Anrede, Wirkung, Tarifgruppe und Gruppenjahre sind keine Kombinationsfelder, sondern aufklappbare Listenfelder. Selbstverständlich ohne die Möglichkeit einer Mehrfachselektion vorzugeben. Die Werte („Herrn“ und „Frau“ und so weiter) wurden fest in die „Listeneinträge“ aufgenommen.

Beachten Sie, dass Sie nach dem Verschieben der Kontrollfelder oder Hinzufügen von neuen Steuerelementen die Reihenfolge anpassen sollten.

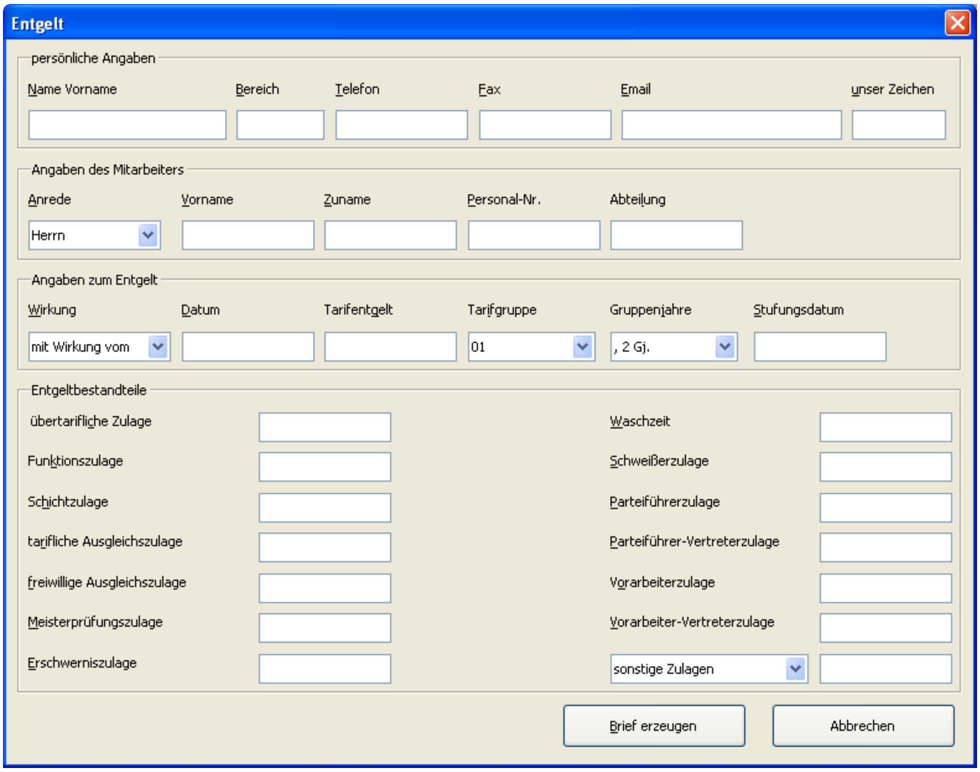


Abbildung 11.182 Der fertige Dialog

Der Dialog wird wie in Kapitel 2 beschrieben gestartet. Die Variable „Dlg“ ist global deklariert. Vor dem Start werden die einzelnen Listenfelder auf ihren jeweils ersten Eintrag gesetzt.

```
   Public Dlg As Object

Sub MaskeStart

   DialogLibraries.LoadLibrary("Standard")
   Dlg = CreateUnoDialog(DialogLibraries.Standard.dlgEntgelt)

   With Dlg
      .getControl("lstAnrede").selectItem _
         (Dlg.getControl("lstAnrede").getItem(0), True)
```

```
      .getControl("lstWirkung").selectItem _
         (Dlg.getControl("lstWirkung"). getItem(0), True)
      .getControl("lstTarifgruppe").selectItem _
         (Dlg.getControl("lstTarifgruppe").getItem(0), True)
      .getControl("lstGruppenjahre").selectItem _
         (Dlg.getControl("lstGruppenjahre").getItem(0), True)
      .getControl("lstZulagen").selectItem _
         (Dlg.getControl("lstZulagen").getItem(0), True)
   End With
   Dlg.Execute()

End Sub
```

Die Voraussetzungen der Eingabe werden überprüft. Das Textfeld „txtTarifentgelt“ ist ein Pflichtfeld, muss also ausgefüllt werden. Dabei entspricht keine Eingabe dem Wert 0. Da die Inhalte der Datumsangaben mit der Funktion `CDateFromIso` in ein „vernünftiges“ Datum konvertiert werden, muss entweder auch bei ihnen sichergestellt werden, dass der Benutzer ein (sinnvolles) Datum eingetragen hat, oder es muss beim Eintragen in das Dokument überprüft werden, ob das Feld leer ist. Sonst liefert `CDateFromIso` einen Fehler:

```
   If Dlg.getControl("txtTarifentgelt").Value = 0 Then
      MsgBox "Bitte geben Sie einen Wert ein!"
      Dlg.getControl("txtTarifentgelt").setFocus()
      Exit Sub
   ElseIf Dlg.getControl("txtDatum").getdate() = 0 Then
      MsgBox "Bitte geben Sie ein Datum ein!"
      Dlg.getControl("txtDatum").setFocus()
      Exit Sub
   ElseIf Dlg.getControl("txtStufungsdatum").getdate() = 0 Then
      MsgBox "Bitte geben Sie ein Datum ein!"
      Dlg.getControl("txtStufungsdatum").setFocus()
      Exit Sub
   End If
```

Die Variablen werden deklariert, der Zeilenabstand für das Dokument festgelegt:

```
   Dim LS As New com.sun.star.style.LineSpacing
```

```
LS.Mode = com.sun.star.style.LineSpacingMode.MINIMUM
LS.Height = 150
```

Ein neues Dokument wird erstellt, auf dem eine Tabelle mit acht Zeilen und zwei Spalten für den Briefkopf erzeugt wird:

```
strUrl = "private:factory/swriter"
objDoc = StarDesktop.loadComponentFromURL _
   (strUrl, "_blank", 0, docArgs())
objTextCursor = objDoc.Text.createTextCursor()
objTabelle = objDoc.createInstance("com.sun.star.text.TextTable")
objTabelle.initialize(8, 2)
objDoc.Text.insertTextContent(objTextCursor, objTabelle, False)
```

Die Linienformatierung wird entfernt:

```
objTabelle.TableBorder.TopLine.InnerLineWidth = 0
objTabelle.TableBorder.TopLine.OuterLineWidth = 0
objTabelle.TableBorder.LeftLine.OuterLineWidth = 0
objTabelle.TableBorder.BottomLine.OuterLineWidth = 0
objTabelle.TableBorder.RightLine.OuterLineWidth = 0
objTabelle.TableBorder.HorizontalLine.OuterLineWidth = 0
objTabelle.TableBorder.VerticalLine.OuterLineWidth = 0
```

Hier scheint ein Fehler vorzuliegen, da nichts passiert!

Ein Cursorobjekt wird erstellt, das auf die Tabelle zugreift und sie formatiert:

```
objZellCursor = objTabelle.getCellRangeByName("A1:B8")
With objZellCursor
   .ParaStyleName = "Standard"
   .CharFontName = "Arial"
   .CharHeight = 11
   .CharUnderline = com.sun.star.awt.FontUnderline.NONE
   .CharWeight = com.sun.star.awt.FontWeight.NORMAL
   .CharPosture = com.sun.star.awt.FontSlant.NONE
   .ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
```

```
    .ParaLineSpacing = LS
    .ParaBottomMargin = 0
End With
```

Einige Sonderfälle werden einzeln behandelt:

```
objZellCursor = objTabelle.createCursorByCellName("B1")
objZellCursor.CharHeight = 8
objZellCursor = objTabelle.createCursorByCellName("A2")
objZellCursor.CharHeight = 8
objZellCursor = objTabelle.createCursorByCellName("B4")
objZellCursor.CharHeight = 8
objZellCursor = objTabelle.createCursorByCellName("A7")
objZellCursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
```

¶	René·Martin¶ IP¶ ¶ compurem·GmbH¶ Häberlstr.·13¶ D-80337·München¶ Tel.+49-89-54-40-44-007¶ Fax.+49-89-54-40-44-008¶ Rene.Martin@compurem.de@compurem.de¶ ¶ ¶ ¶
compurem·GmbH·/·Häberlstr.·13·/·80337·München¶	¶
Herrn¶ Lothar·Zeyer¶ Personalnr.·12345¶ ¶ IP¶ ¶	¶
¶	Unser·Zeichen¶
28.06.2005¶	rem¶
¶	¶
Entgeltänderung¶	¶
¶	¶

¶

Abbildung 11.183 Der Briefkopf – eine Tabelle

Da sehr viel Text eingetragen wird, hinter dem sich jeweils ein Absatzwechsel befindet, wird nicht nur mit einem Textcursor gearbeitet:

```
   objZelle = objTabelle.getCellByPosition(1, 0)
   objTextCursor = objZelle.createTextCursor()
```

sondern das Schreiben wird in eine eigene Prozedur ausgelagert:

```
Sub TextUndEnter(strText As String)
   Const ABSATZ As Integer = 0

   objTextCursor.String = strText
   objTextCursor.goRight(Len(strText), False)
   objDoc.Text.insertControlCharacter(objTextCursor, ABSATZ, False)
End Sub
```

Sie kann nun für die Texte aufgerufen werden, die in die einzelnen Zellen der Tabelle geschrieben werden. Die Zelle B1:

```
   Call TextUndEnter(Dlg.getControl("txtNameVorname").getText())
   Call TextUndEnter(Dlg.getControl("txtBereich").getText())
   Call TextUndEnter("")

   Call TextUndEnter("compurem GmbH")
   Call TextUndEnter("Häberlstr. 13")
   Call TextUndEnter("D-80337 München")
   Call TextUndEnter("Tel.+49-89-54-40-44-" & _
      Dlg.getControl("txtTelefon").getText())
   Call TextUndEnter("Fax.+49-89-54-40-44-" & _
      Dlg.getControl("txtFax").getText())

   Call TextUndEnter(Dlg.getControl("txtEmail").getText() & _
      "@compurem.de")
   Call TextUndEnter("")
   Call TextUndEnter("")
```

Wird ein leerer Absatz benötigt, dann wird an die Prozedur „TextUndEnter" eine leere Zeichenkette übergeben. Die Anschrift wird in A2 eingetragen:

```
   objZelle = objTabelle.getCellByPosition(0, 1)
   objTextCursor = objZelle.createTextCursor()
```

```
objTextCursor.String = "compurem GmbH / Häberlstr. 13 / 80337 München"

objZelle = objTabelle.getCellByPosition(0, 2)
objTextCursor = objZelle.createTextCursor()
Call TextUndEnter(Dlg.getControl("lstAnrede").selectedItem)
Call TextUndEnter(Dlg.getControl("txtVorname").getText() & " " & _
   Dlg.getControl("txtZuname").getText())
Call TextUndEnter("Personalnr. " & _
   Dlg.getControl("txtPersNr").getText())
Call TextUndEnter("")
Call TextUndEnter(Dlg.getControl("txtAbteilung").getText())
```

Darunter die übrigen Informationen und das aktuelle Datum (`Date`):

```
objZelle = objTabelle.getCellByPosition(1, 3)
objTextCursor = objZelle.createTextCursor()
objTextCursor.String = "Unser Zeichen"
objZelle = objTabelle.getCellByPosition(0, 4)
objTextCursor = objZelle.createTextCursor()
objTextCursor.String = Date
objZelle = objTabelle.getCellByPosition(1, 4)
objTextCursor = objZelle.createTextCursor()
objTextCursor.String = Dlg.getControl("txtUnserZeichen").getText()
objZelle = objTabelle.getCellByPosition(0, 6)
objTextCursor = objZelle.createTextCursor()
objTextCursor.String = "Entgeltänderung"
```

Für den Fließtext wird die „Standardschrift“ eingeschaltet:

```
objTextCursor = objDoc.Text.createTextCursor()
objTextCursor.gotoEnd(False)
With objTextCursor
   .ParaStyleName = "Standard"
   .CharFontName = "Arial"
   .CharHeight = 11
   .CharUnderline = com.sun.star.awt.FontUnderline.NONE
```

```
      .CharWeight = com.sun.star.awt.FontWeight.NORMAL
      .CharPosture = com.sun.star.awt.FontSlant.NONE
      .ParaAdjust = com.sun.star.style.ParagraphAdjust.LEFT
      .ParaLineSpacing = LS
      .ParaBottomMargin = 0
   End With
```

Es folgt erneut Text:

```
   Call TextUndEnter("")
   If Dlg.getControl("lstAnrede").selectedItem = "Herrn" Then
      Call TextUndEnter("Sehr geehrter Herr " & _
         Dlg.getControl("txtZuname").getText() & ",")
   Else
      Call TextUndEnter("Sehr geehrte Frau " & _
         Dlg.getControl("txtZuname").getText() & ",")
   End If
   Call TextUndEnter("")
   Call TextUndEnter("Wir freuen uns, Ihnen mitteilen zu können " & _
      dass sich Dlg.getControl("lstWirkung").selectedItem & " " & _
      CDateFromIso(Dlg.getControl("txtDatum").getDate()) & _
      " unter Vorwegnahme des Wechsels der Jahre Ihr Monatsentgelt " & _
      "(brutto) wie folgt errechnet:")
   Call TextUndEnter("")
```

Beachten Sie, dass für die Anrede zwischen Frauen und Männern unterschieden werden muss.

Aus dem Datumsfeld „txtDatum“ wird das eingegebene Datum nicht mit getText, sondern mit `getDate` ausgelesen. Um es in ein „vernünftiges“ Datum zu verwandeln, muss es mit der Funktion `CDateFromIso` konvertiert werden. Beachten Sie, dass dies fehlschlägt, wenn der Benutzer kein Datum eingegeben hat!

Anschließend wird eine neue Tabelle erzeugt:

```
   objTabelle = objDoc.createInstance("com.sun.star.text.TextTable")
   objTabelle.initialize(2, 2)
```

```
objDoc.Text.insertTextContent(objTextCursor, objTabelle, False)
```

Erstaunlicherweise muss man das Tabellenobjekt mit `createInstance` erneut erzeugen. Die Tabelle wird wie oben formatiert. Zusätzlich wird sie auf der Seite zentriert und etwas verkleinert:

```
objTabelle.HoriOrient = com.sun.star.text.HoriOrientation.CENTER
objTabelle.RelativeWidth = 75
```

Da die Anzahl der Tabellenzeilen unterschiedlich ist, gibt es verschiedene Strategien:

- Man könnte mit einem Zähler die 14 Textfelder durchlaufen und überprüfen, ob sich Zahlenwerte darin befinden. Die Tabelle wird mit so vielen Zeilen erzeugt.
- Man könnte eine maximale Tabelle (hier: 14 Zeilen plus eine für den Kopf und eine für die Summe) erzeugen und anschließend die Zeilen löschen, falls nichts eingegeben wurde.
- Ich habe mich für die Lösung entschieden, für jeden Wert, der eingetragen wurde, eine neue Zeile zu generieren. Da OOo mit `insertByIndex` die neue Zeile allerdings oberhalb des Indexwertes einträgt, muss die Liste entweder von unten nach oben durchlaufen werden, oder ich füge jeweils über der letzten Zeile eine neue Zeile ein. Für diese Methode habe ich mich dann auch entschieden.

Und so werden die Informationen eingetragen:

```
objZelle = objTabelle.getCellByPosition(0, 0)
objTextCursor = objZelle.createTextCursor()
objTextCursor.String = "Tarifentgelt E " & _
   Dlg.getControl("lstTarifgruppe").selectedItem & _
   Dlg.getControl("lstGruppenjahre").selectedItem
objTabelle.getCellByName("B1").Value = _
   Dlg.getControl("txtTarifentgelt").Value

If Dlg.getControl("txtUebertariflicheZulage").Value <> 0 Then
   objTabelle.getRows().insertByIndex(objTabelle.getRows(). _
      Count -1, 1)
   objTabelle.getCellByPosition(0, objTabelle.getRows(). _
      Count -2).String = "Übertarifliche Zulage"
   objTabelle.getCellByPosition(1, objTabelle.getRows(). _
      Count -2).Value = _
      Dlg.getControl("txtUebertariflicheZulage").Value
   objTabelle.getCellByPosition(1, objTabelle.getRows(). _
```

```
        Count -2).NumberFormat = 106
    End If
...
    If Dlg.getControl("txtZulagen").Value <> 0 Then
        objTabelle.getRows().insertByIndex(objTabelle.getRows(). _
            Count -1, 1)
        objTabelle.getCellByPosition(0, objTabelle.getRows(). _
            Count -2).String = _
            Dlg.getControl("lstZulagen").selectedItem
        objTabelle.getCellByPosition(1, objTabelle.getRows(). _
            Count -2).Value = _
            Dlg.getControl("txtZulagen").Value
        objTabelle.getCellByPosition(1, objTabelle.getRows(). _
            Count -2).NumberFormat = 106
    End If
```

¶
Wir freuen uns, Ihnen mitteilen zu können, dass sich mit Wirkung vom 01.06.2005 unter Vorwegnahme des Wechsels der Jahre Ihr Monatsentgelt (brutto) wie folgt errechnet:¶
¶

Tarifentgelt E 04, 2 Gj.¶	500
Übertarifliche Zulage¶	50
Funktionszulage¶	20
Freiwillige Ausgleichszulage¶	10
Erschwerniszulage¶	5
sonstige Zulagen¶	5
Summe¶	590

¶
Als Stufungsdatum wurde aufgrund der tariflichen Regelung der 01.01.2005 festgelegt.¶

Abbildung 11.184 Die zweite Tabelle mit den Daten

Wurden neben der „Hauptzeile“ keine weiteren wichtigen Informationen eingetragen, dann kann die letzte Zeile gelöscht werden:

```
    If objTabelle.getRows().Count = 2 Then
        objTabelle.getRows().removeByIndex(1, 1)
```

Falls mindestens eine Zeile hinzugefügt wird, wird die Summe ermittelt. Sie muss in der Form

```
=sum(<B1>|<B2>|<B3>|<B4>|<B5>|<B6>|<B7>|<B8>)
```

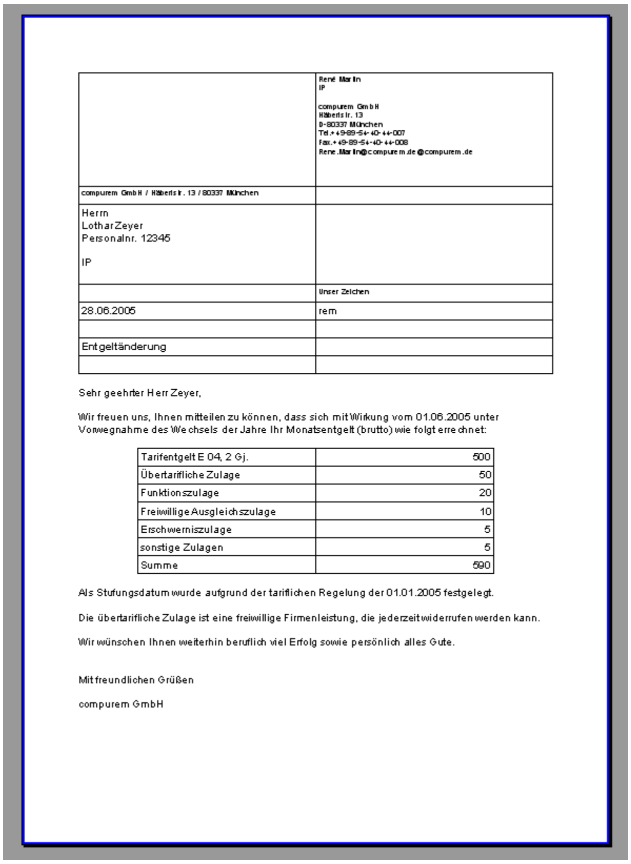
in der letzten Zelle stehen. Dafür wird mit Hilfe einer Schleife die Zeichenkette zusammengesetzt und mit der Methode `setFormula` in die letzte Zelle geschrieben:

```
Else
   strFormel = "=sum("
   For intZeilen = 1 To objTabelle.getRows().Count - 1
      strFormel = strFormel & "<B" & intZeilen & ">|"
   Next
   strFormel = Left(strFormel, Len(strFormel) - 1) & ")"
   objTabelle.getCellByPosition(0, objTabelle.getRows(). _
      Count -1).String = "Summe"
   objTabelle.getCellByPosition(1, objTabelle.getRows(). _
      Count -1).setFormula(strFormel)
End If
```

Der übrige Text am Ende des Briefes stellt nun keine Schwierigkeit mehr dar:

```
objTextCursor = objDoc.Text.createTextCursor()
objTextCursor.gotoEnd(False)
Call TextUndEnter("")
Call TextUndEnter _
("Als Stufungsdatum wurde aufgrund der tariflichen Regelung der " & _
   CDateFromIso(Dlg.getControl("txtStufungsdatum").getDate()) & " _
      festgelegt.")
Call TextUndEnter("")
If Dlg.getControl("txtUebertariflicheZulage").Value <> 0 Then
   Call TextUndEnter("Die übertarifliche Zulage ist eine " & _
  "freiwillige Firmenleistung, die jederzeit widerrufen werden kann.")
   Call TextUndEnter("")
End If
Call TextUndEnter("Wir wünschen Ihnen weiterhin beruflich viel " & _
   "Erfolg sowie persönlich alles Gute.")
Call TextUndEnter("")
Call TextUndEnter("")
```

```
Call TextUndEnter("Mit freundlichen Grüßen")
Call TextUndEnter("")
Call TextUndEnter("compurem GmbH")

Dlg.endExecute()
```



	René Martin IP compurem GmbH Häberlstr. 13 D-80337 München Tel.+49-89-54-40-44-007 Fax.+49-89-54-40-44-008 Rene.Martin@compurem.de@compurem.de
compurem GmbH / Häberlstr. 13 / 80337 München	
Herrn Lothar Zeyer Personalnr. 12345 IP	
	Unser Zeichen
28.06.2005	rem
Entgeltänderung	

Sehr geehrter Herr Zeyer,

Wir freuen uns, Ihnen mitteilen zu können, dass sich mit Wirkung vom 01.06.2005 unter Vorwegnahme des Wechsels der Jahre Ihr Monatsentgelt (brutto) wie folgt errechnet:

Tarifentgelt E 04, 2 Gj.	500
Übertarifliche Zulage	50
Funktionszulage	20
Freiwillige Ausgleichszulage	10
Erschwerniszulage	5
sonstige Zulagen	5
Summe	590

Als Stufungsdatum wurde aufgrund der tariflichen Regelung der 01.01.2005 festgelegt.

Die übertarifliche Zulage ist eine freiwillige Firmenleistung, die jederzeit widerrufen werden kann.

Wir wünschen Ihnen weiterhin beruflich viel Erfolg sowie persönlich alles Gute.

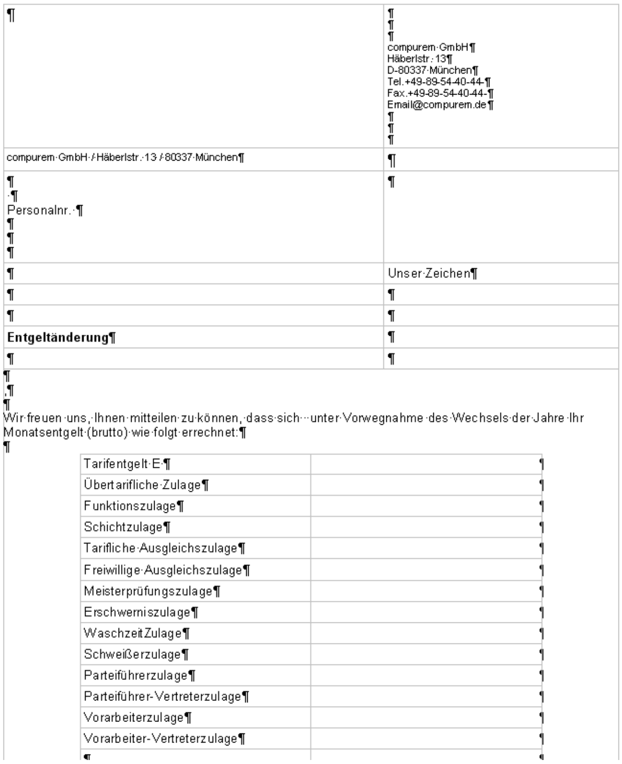
Mit freundlichen Grüßen

compurem GmbH

Abbildung 11.185 Der fertige Brief

11.2 Der Entgeltbrief II

Wie das obere Beispiel gezeigt hat, ist es sehr mühsam und aufwändig, sämtliche Texte in ein Dokument zu schreiben, Tabellen und andere Textelemente einzufügen und sämtliche Formatierungen per Programmierung vorzunehmen. Eleganter und effektiver ist es, das „Gerüst“ in einem Writer-Dokument zu erstellen, in das Dokument Textmarken einzubauen, die angesprungen und mit dem entsprechenden Text gefüllt werden.



¶	¶ ¶ ¶ compurem·GmbH¶ Häberlstr.·13¶ D-80337·München¶ Tel.+49-89-54-40-44-¶ Fax.+49-89-54-40-44-¶ Email@compurem.de¶ ¶ ¶ ¶
compurem·GmbH·/·Häberlstr.·13·/·80337·München¶	¶
¶ ·¶ Personalnr.·¶ ¶ ¶ ¶	¶
¶	Unser·Zeichen¶
¶	¶
¶	¶
Entgeltänderung¶	¶
¶	¶

¶
,¶
¶
Wir·freuen·uns,·Ihnen·mitteilen·zu·können,·dass·sich···unter·Vorwegnahme·des·Wechsels·der·Jahre·Ihr Monatsentgelt·(brutto)·wie·folgt·errechnet:¶
¶

Tarifentgelt·E·¶	¶
Übertarifliche·Zulage¶	¶
Funktionszulage¶	¶
Schichtzulage¶	¶
Tarifliche·Ausgleichszulage¶	¶
Freiwillige·Ausgleichszulage¶	¶
Meisterprüfungszulage¶	¶
Erschwerniszulage¶	¶
WaschzeitZulage¶	¶
Schweißerzulage¶	¶
Parteiführerzulage¶	¶
Parteiführer-Vertreterzulage¶	¶
Vorarbeiterzulage¶	¶
Vorarbeiter-Vertreterzulage¶	¶

Abbildung 11.186 Das „Brief-Gerüst“

Im Dialog tragen sämtliche Steuerelemente einen Namen, die mit einer Textmarke im Dokument korrespondieren, also der Inhalt von „txtNameVorname“ wird an die Stelle „NameVorname“ geschrieben, der Inhalt von „txtTarifentgelt“ an „Tarifentgelt“, der ausgewählte Eintrag von „lstWirkung“ an „Wirkung“ und so weiter. So kann man mit einer Schleife über sämtliche Kontrollelemente laufen und deren Inhalt leicht in das Dokument eintragen:

```
   objDatei = ThisComponent
   objTextmarken = objDatei.getBookmarks()
   For intControlZaehler = 0 To UBound(Dlg.Controls)
```

Dazu wird der Name jedes Kontrollelements ausgelesen:

```
      strMarke = Mid(Dlg.Controls(intControlZaehler).Model.Name, 4)
```

Und danach wird überprüft, um welche Art Element es sich handelt:

```
      Select Case Dlg.Controls(intControlZaehler).ImplementationName


      Case "stardiv.Toolkit.UnoEditControl"
         If objTextmarken.hasByName(strMarke) = True Then
            objTextmarke = objTextmarken.getByName(strMarke)
            objTextmarke.Anchor.setString(Dlg.Controls _
               (intControlZaehler).getText())
         End If
```

Während es bei Textfeldern genügt, den Inhalt mit `Text` oder `getText()` auszulesen, muss bei Datumsfeldern nicht nur mit der Methode `getDate()` gearbeitet werden, sondern das Datum muss auch hier wieder mit `CDateFromIso` konvertiert werden.

Beachten Sie, dass diese Datumsfelder entweder ein Datum haben müssen oder man sie auf leere Eingaben überprüfen muss:

```
      Case "stardiv.Toolkit.UnoDateFieldControl"
         If objTextmarken.hasByName(strMarke) = True Then
            objTextmarke = objTextmarken.getByName(strMarke)
            objTextmarke.Anchor.setString(CDateFromIso( _
               Dlg.Controls(intControlZaehler).getDate()))
         End If
      Case "stardiv.Toolkit.UnoNumericFieldControl"
         If objTextmarken.hasByName(strMarke) = True Then
            objTextmarke = objTextmarken.getByName(strMarke)
            objTextmarke.Anchor.setString(Dlg.Controls _
               (intControlZaehler).getValue())
         End If
```

Der Inhalt der Währungsfelder wird formatiert und in einer Variablen („dblSumme“) gespeichert. Man könnte – analog zum ersten Beispiel – überflüssige Zeilen löschen.

```
      Case "stardiv.Toolkit.UnoCurrencyFieldControl"
         If objTextmarken.hasByName(strMarke) = True Then
            objTextmarke = objTextmarken.getByName(strMarke)
            If Dlg.Controls(intControlZaehler).getValue() > 0 Then
               dblSumme = dblSumme + _
                  Dlg.Controls(intControlZaehler).getValue()
               objTextmarke.Anchor.setString(Format _
                  (Dlg.Controls(intControlZaehler).getValue(), _
                   "#,##0.00") & " €")
            End If
         End If
```

Ebenso wird die Auswahl der Listenfelder ausgelesen und eingetragen:

```
      Case "stardiv.Toolkit.UnoListBoxControl"
         If objTextmarken.hasByName(strMarke) = True Then
            objTextmarke = objTextmarken.getByName(strMarke)
            objTextmarke.Anchor.setString(Dlg.Controls _
               (intControlZaehler).selectedItem)
         End If
      End Select
   Next
```

An ein weiteres Textfeld („Summe“) wird der Wert der Variablen „dblSumme“ übergeben und formatiert:

```
   objTextmarken.getByName("Summe").Anchor.setString _
      (Format(dblSumme, "#,##0.00") & " €")
```

Da sich die Anrede für Männer und Frauen unterscheidet, wird bei der Textmarke „Briefanrede“ differenziert:

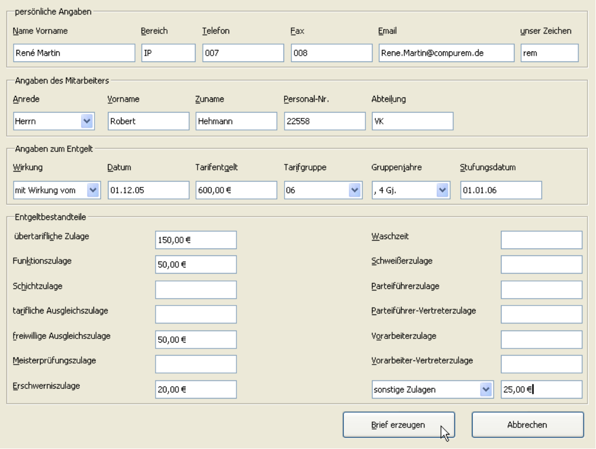
```
   If Dlg.getControl("lstAnrede").selectedItem = "Frau" Then
      strMarke = "Sehr geehrte Frau " & _
         Dlg.getControl("txtZuname").getText
   Else
```

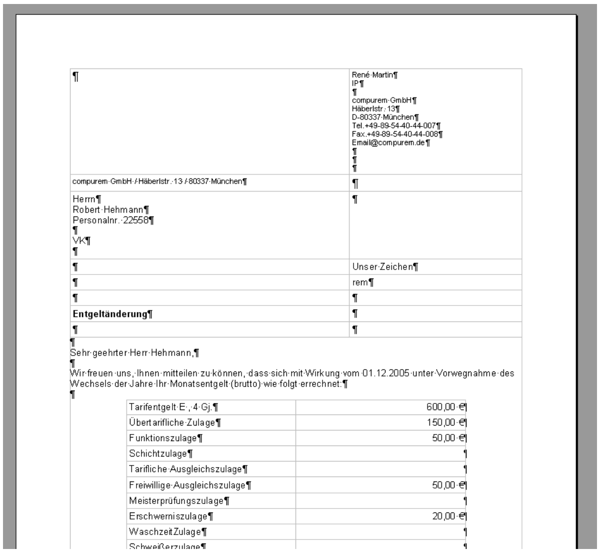
```
      strMarke = "Sehr geehrter Herr " & _
         Dlg.getControl("txtZuname").getText
   End If
   objTextmarke = objTextmarken.getByName("Briefanrede")
   objTextmarke.Anchor.setString(strMarke)
```

Der Dialog wird geschlossen:

```
   Dlg.endExecute()
```

Das Ergebnis ist ähnlich wie im ersten Fall – jedoch ist der Programmieraufwand ungleich geringer:





René Martin
IP

compurem GmbH
Häberlstr. 13
D-80337 München
Tel.+49-89-54-40-44-007
Fax.+49-89-54-40-44-008
Email@compurem.de

compurem GmbH / Häberlstr. 13 / 80337 München

Herrn
Robert Hehmann
Personalnr. 22558

VK

Unser Zeichen
rem

Entgeltänderung

Sehr geehrter Herr Hehmann,

Wir freuen uns, Ihnen mitteilen zu können, dass sich mit Wirkung vom 01.12.2005 unter Vorwegnahme des Wechsels der Jahre Ihr Monatsentgelt (brutto) wie folgt errechnet:

Tarifentgelt E., 4 Gj.	600,00 €
Übertarifliche Zulage	150,00 €
Funktionszulage	50,00 €
Schichtzulage	
Tarifliche Ausgleichszulage	
Freiwillige Ausgleichszulage	50,00 €
Meisterprüfungszulage	
Erschwerniszulage	20,00 €
WaschzeitZulage	
Schweißerzulage	

Abbildung 11.187 Der Dialog und der fertige Brief

Damit der Dialog beim Laden des Dokumentes angezeigt wird, muss noch das entsprechende Makro an das richtige Ereignis gebunden werden:

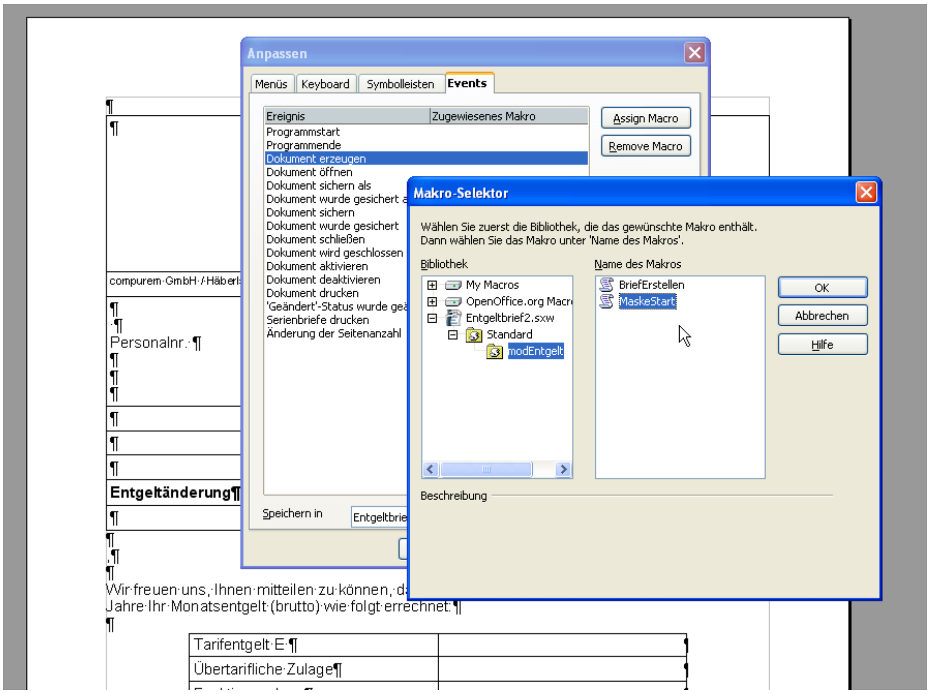


Abbildung 11.188 An das Ereignis „Dokument erzeugen“ wird das Makro gebunden.

11.3 Aufhebungsvertrag

Eine dritte Variante, bei der Texte für Formulare erstellt werden können, sieht die folgende Lösung vor.

In einem Modul werden mehrere „Textbausteine“ definiert; dabei liegt jeder Baustein in zwei „Schreibweisen“ vor: einmal als angezeigter (Kurz-)Text und dann als beschreibender Informationstext:

```
   Const ANZAHL_TB As Integer = 32
   Dim strTB(ANZAHL_TB) As String
   Dim strK(ANZAHL_TB) As String
Sub DialogStart

   strTB(0) = "§ 1 Beendigung_ohne_Kdg"
```

```
   strTB(1) = "§ 1 Beendigung_mit_Kdg_Frist"

   strTB(2) = "§ 2 Entgelt_Urlaub_Abgeltung"
   strTB(3) = "§ 2 Entgelt_Urlaub_Einbringung"
   strTB(4) = "§ 2 Entgelt_Urlaub_Freistellung1"
   strTB(5) = "§ 2 Entgelt_Urlaub_Freistellung2"
...
   strK(0) = "Paragraph 1, Beendigung ohne Einhaltung einer Kündigungsfrist"
   strK(1) = "Paragraph 1, Beendigung unter Einhaltung der ordentlichen Kündi-
gungsfrist"

   strK(2) = "Paragraph 2, wenn der Mitarbeiter unter Anrechnung der zustehenden
Urlaubsansprüche freigestellt wird"
   strK(3) = "Paragraph 2, wenn der noch zustehende Urlaubsanspruch wegen vorzei-
tiger Beendigung des Anstellungsverhältnisses abgegolten wird"
   strK(4) = "Paragraph 2, wenn der noch zustehende Urlaubsanspruch bis zur Been-
digung des Anstellungsverhältnisses einzubringen ist"
   strK(5) = "Paragraph 2, wenn der Mitarbeiter unter Anrechnung der zustehenden
Urlaubsansprüche und vorhandenen Zeitguthaben freigestellt wird"
```

Der Dialog wird gestartet, und ein Listenfeld wird mit den Textbausteinen gefüllt.

```
   For intZaehler = 0 To ANZAHL_TB
      Dlg.getControl("lstLinks").addItem(strTB(intZaehler), intZaehler)
   Next
```

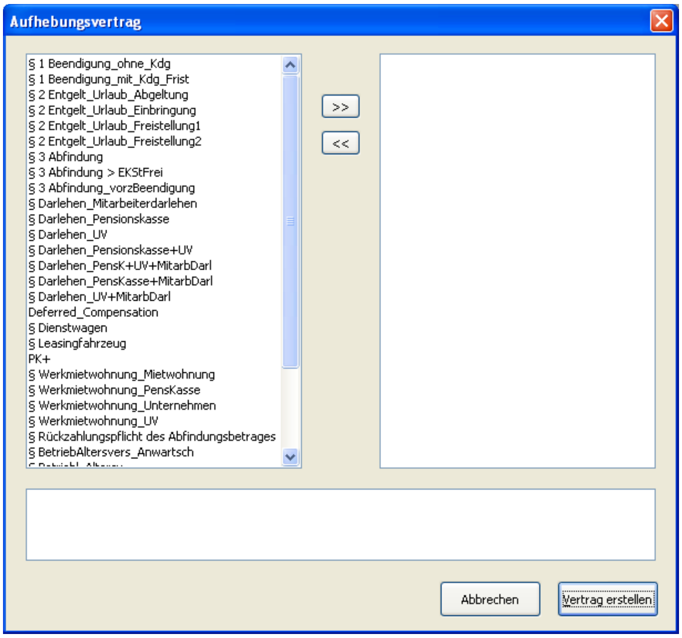


Abbildung 11.189 In der linken Spalten befinden sich die „Textbaustein-Namen".

Wählt der Benutzer im linken (analog im rechten) Fenster einen Baustein aus, wird der zugehörige Info-Text im unteren Textfeld angezeigt:

```
Sub lstLinks_Selected
   If Dlg.getControl("lstLinks").selectedItemPos > -1 Then
      For intZaehler = 0 To ANZAHL_TB
         If Dlg.getControl("lstLinks").selectedItem = _
            strTB(intZaehler) Then
            Dlg.getControl("txtK").setText(strK(intZaehler))
            Exit For
         End If
      Next

   End If
End Sub
```

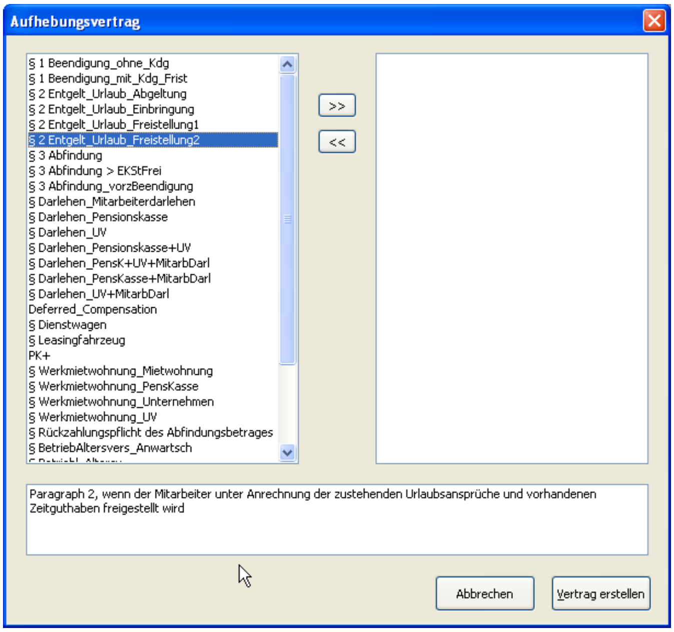


Abbildung 11.190 Der zugehörige Text wird angezeigt.

Über die Schaltflächen oder auch per Doppelklick auf einen der Einträge („beim Auslösen“ bzw. „Maustaste gedrückt“) wird ein Eintrag vom linken Listenfeld ins rechte geschoben und vice versa:

```
Sub VonLinksNachRechts
   Dim intAnzahlRechts As Integer
   If Dlg.getControl("lstLinks").selectedItemPos > -1 Then
      intAnzahlRechts = Dlg.getControl("lstRechts").ItemCount
      Dlg.getControl("lstRechts").addItem _
         (Dlg.getControl("lstLinks").selectedItem, intAnzahlRechts)
      Dlg.getControl("lstLinks").removeItems _
         (Dlg.getControl("lstLinks").selectedItemPos, 1)
      Dlg.getControl("txtK").setText("")
   Else
      MsgBox "Bitte markieren Sie einen Eintrag!"
   End If
```

```
End Sub
```

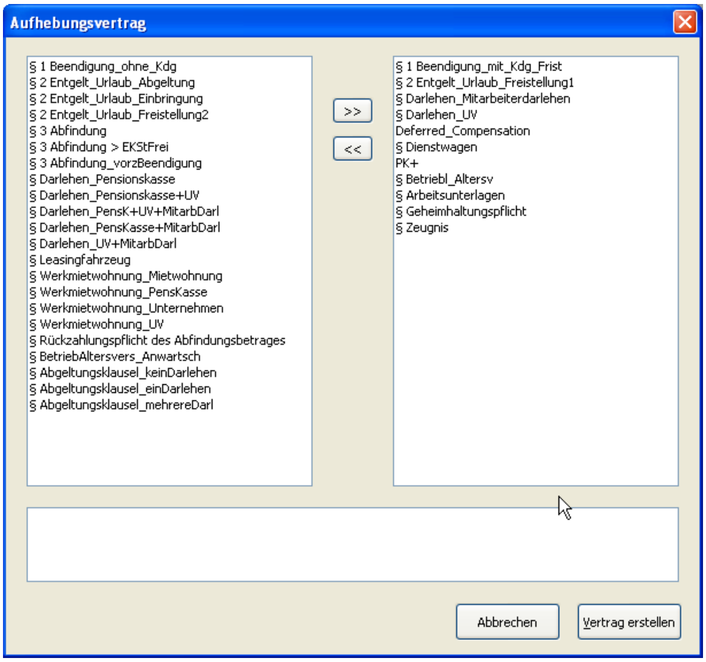


Abbildung 11.191 Aus den Bausteinen der rechten Spalte wird der Vertrag generiert.

Wenn schließlich der Anwender die gewünschten Einträge zusammengestellt hat, kann daraus ein Brief erstellt werden.

Intelligente Formulare

Die Aufgabe ist denkbar einfach: In einer großen Firma möchten die Mitarbeiter regelmäßig Bilder auf Papier ausgedruckt bekommen. Deshalb soll in einer Präsentation eine „Vorschau“ aller Bilder erstellt werden, die sich in einem bestimmten Ordner befinden. Diese können von dem Mitarbeiter ausgewählt und „bestellt“ werden.

12Eine Bilderdatenbank in Impress

12.1 Ordnerauswahl

Um den Ordner auszuwählen, kann der PickFolder-Dialog verwendet werden. Der Ordner, der vom Benutzer angeklickt wird, wird an eine Variable übergeben und auf Gültigkeit überprüft:

```
objVerzeichnisDlg = _
   CreateUnoService("com.sun.star.ui.dialogs.FolderPicker")

If objVerzeichnisDlg.execute() Then
   strPfad = objVerzeichnisDlg.Directory
End If

If Right(strPfad, 1) <> "/" Then
    strPfad = strPfad & "/"
End If

If Dir(strPfad, 16) = "" Then
   MsgBox "Das von Ihnen angegebene Verzeichnis ist nicht gültig!"
ElseIf strPfad = "/" Then
   MsgBox "Sie haben nichts ausgewählt!"
Else
```

Abbildung 12.192 Der Auswahldialog für die Ordner

Falls die Voraussetzungen erfüllt sind, wird eine neue, leere Präsentation geöffnet und für jedes Bild eine neue Seite erzeugt. Jedes Bild wird auf der Seite platziert:

```
      strUrl = "private:factory/simpress"
      objDoc = StarDesktop.loadComponentFromURL(strUrl, _
         "_blank", 0, docArgs())
      ' -- neue Präsentation

         Punkt.x = 1000
         Punkt.y = 1000
         Groesse.Width = 24000
         Groesse.Height = 18000

   i = 2
   strDir="Dateien:"
   strWert = Dir(strPfad, 0)

   Do Until strWert = ""
      If strWert <> "." and strWert <> ".." Then
```

```
        If (GetAttr(strPfad & strWert) AND 0) = 0 then
          Select Case LCase(Right(strWert, 3))
          Case "jpg", "gif", "tif", "png", "bmp"

          objBlatt = objDoc.getDrawPages().insertNewByIndex(i)
          objGrafik = objDoc.createInstance _
            ("com.sun.star.drawing.GraphicObjectShape")
          objGrafik.Size = Groesse
          objGrafik.Position = Punkt
          objGrafik.GraphicURL = strPfad & strWert
          objBlatt.add(objGrafik)
          i = i + 1
          objBlatt.setName(strWert)
          End Select
        End If
      End If
  strWert = Dir$
  Loop
```

Es wird überprüft, ob Bilddateien in dem Ordner gefunden wurden:

```
      If objDoc.getDrawPages().getCount() = 1 Then
        MsgBox "Es wurden keine Bilder im Ordner" & _
        Chr(13) & strPfad & chr(13) & "gefunden."
      Else
```

Falls ja, wird unter der neuen Datei eine Makrobibliothek erstellt. In sie wird eine Bibliothek (und damit die entsprechenden Makros) der aktuellen Datei kopiert:

```
        objBibliNeu = objDoc.BasicLibraries.createLibrary("BilderDB")
        ThisComponent.BasicLibraries.loadLibrary("Bilder")
        objBibliAlt = ThisComponent.BasicLibraries.getByName("Liste")
        objBibliNeu.insertByName("modListe", _
          objBibliAlt.getByName("modListe")
```

Nun kann auf dem letzten Blatt eine Schaltfläche erstellt werden:

```
        objDoc.getDrawPages().remove(objDoc. _
```

```
   getDrawPages().getByIndex(0))
objBlatt = objDoc.getDrawPages().insertNewByIndex(i)
objBlatt.Name = "bestellen"
objForm = objDoc.createInstance _
   ("com.sun.star.form.component.Form")
objForm.Name = "Auswahl"
objBlatt.Forms.insertByIndex(0, objForm)



Punkt.x = 1000
Punkt.y = 1000
Groesse.Width = 5000
Groesse.Height = 2000
objShape = objDoc.createInstance _
   ("com.sun.star.drawing.ControlShape")
objShape.Size = Groesse
objShape.Position = Punkt

objKontroll = objDoc.createInstance _
   ("com.sun.star.form.component.CommandButton")
objKontroll.Name = "K0"
objKontroll.Label = "Liste anzeigen"
objShape.Control = objKontroll

objForm.insertByIndex(0, objKontroll)
objBlatt.add(objShape)
```

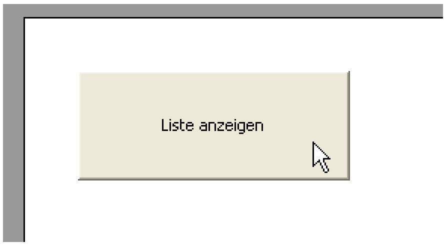


Abbildung 12.193 Die Schaltfläche auf der letzten Folie

Ebenso kann auf jeder Seite ein Kontrollkästchen eingefügt werden:

```
Punkt.x = 1000
Punkt.y = 20000
Groesse.Width = 10000
Groesse.Height = 1000

For i = 0 To objDoc.getDrawPages().getCount() - 2
   objShape = objDoc.createInstance _
      ("com.sun.star.drawing.ControlShape")
   objShape.Size = Groesse
   objShape.Position = Punkt

   objKontroll = objDoc.createInstance _
      ("com.sun.star.form.component.CheckBox")
   objKontroll.Name = "K" & i
   objKontroll.Label = "bestellen"
   objShape.Control = objKontroll
   objForm = objDoc.createInstance _
      ("com.sun.star.form.component.Form")
   objForm.Name = "Auswahl"
   objDoc.getDrawPages().getByIndex(i).Forms.insertByIndex(0, _
      objForm)
   objForm.insertByIndex(0, objKontroll)
   objDoc.getDrawPages().getByIndex(i).add(objShape)
```

```
        Next
    End If
```

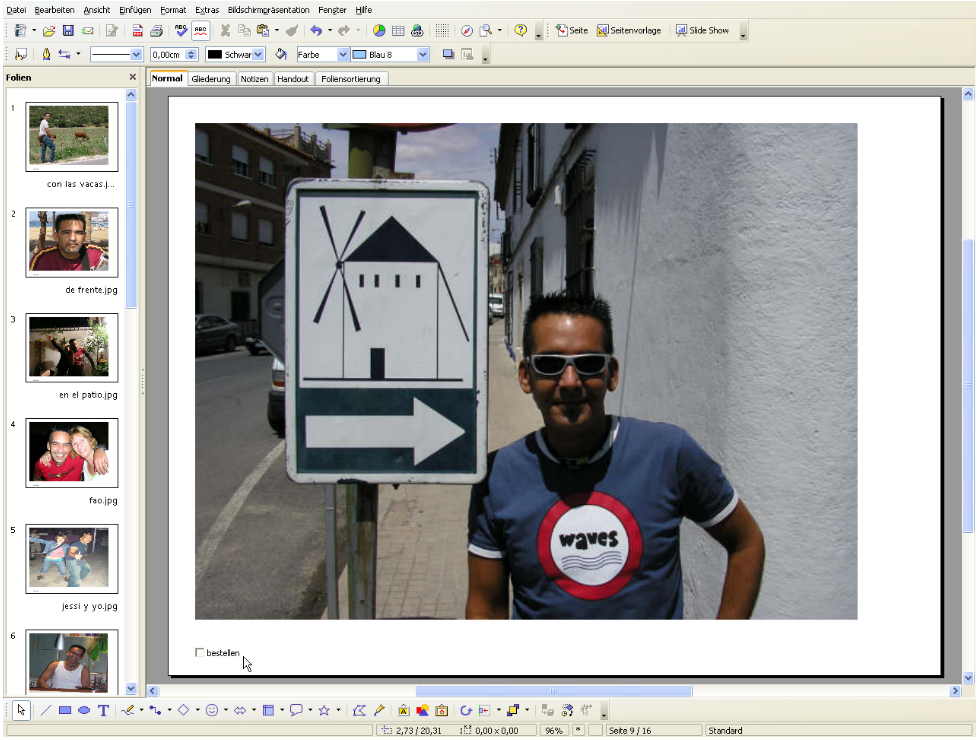


Abbildung 12.194 Die fertige Präsentation

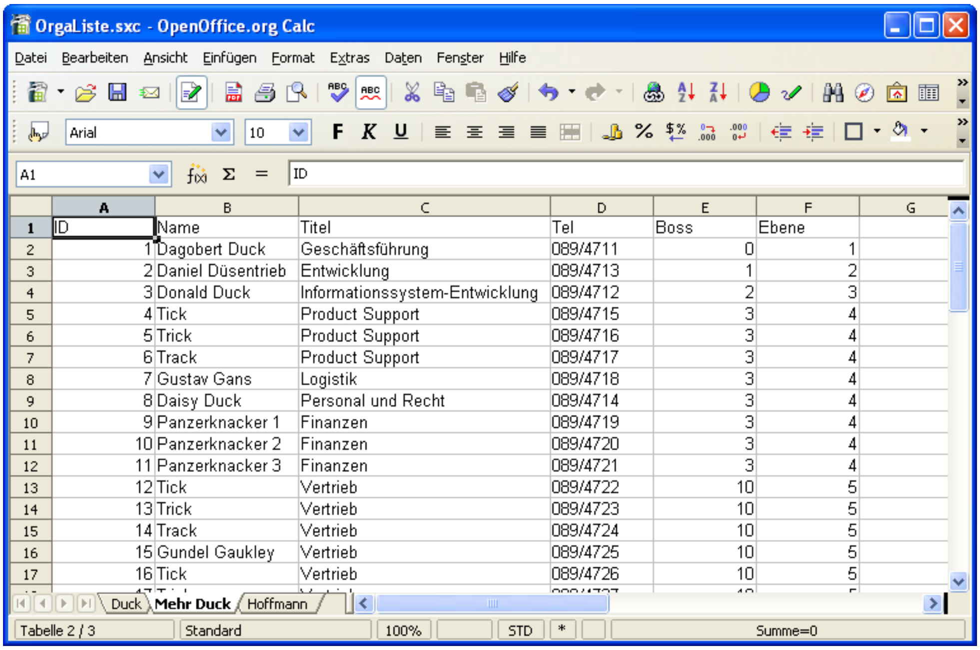
Eine Bilderdatenbank in Impress

In diesem Kapitel wird anhand eines Beispiels beschrieben, wie man von Draw oder Impress auf Calc zugreifen kann, Daten auslesen und in einer Zeichnung verarbeiten kann.

13 Ein Organigramm-Assistent

13.1 Voraussetzung

In einer Tabelle befindet sich eine Reihe von Daten – die Mitarbeiter einer fiktiven Firma. Jeder Datensatz hat eine eindeutige Nummer, jeder Datensatz verweist auf eine höhere Nummer als übergeordnete Position und gibt die Ebene an, auf der er sich befindet.



	A	B	C	D	E	F	G
1	ID	Name	Titel	Tel	Boss	Ebene	
2	1	Dagobert Duck	Geschäftsführung	089/4711	0	1	
3	2	Daniel Düsentrieb	Entwicklung	089/4713	1	2	
4	3	Donald Duck	Informationssystem-Entwicklung	089/4712	2	3	
5	4	Tick	Product Support	089/4715	3	4	
6	5	Trick	Product Support	089/4716	3	4	
7	6	Track	Product Support	089/4717	3	4	
8	7	Gustav Gans	Logistik	089/4718	3	4	
9	8	Daisy Duck	Personal und Recht	089/4714	3	4	
10	9	Panzerknacker 1	Finanzen	089/4719	3	4	
11	10	Panzerknacker 2	Finanzen	089/4720	3	4	
12	11	Panzerknacker 3	Finanzen	089/4721	3	4	
13	12	Tick	Vertrieb	089/4722	10	5	
14	13	Trick	Vertrieb	089/4723	10	5	
15	14	Track	Vertrieb	089/4724	10	5	
16	15	Gundel Gaukley	Vertrieb	089/4725	10	5	
17	16	Tick	Vertrieb	089/4726	10	5	

Abbildung 13.195 Die Liste der Namen

Zugegeben: Die Ebenennummer wäre überflüssig – man könnte sie durch eine weitere Berechnung herausfinden.

Da der Dialog (`Dlg`) dynamisch ist, das heißt: einige Kontrollfelder werden ein- und ausgeblendet, wird eine Prozedur geschrieben, die dies durchführt:

```
Sub Unsichtbar(strService As String, blnSichtbar As Boolean)
   Dim intZaehler As Integer

   For intZaehler = 0 To UBound(Dlg.Controls)

      If Dlg.Controls(intZaehler).Implementationname = _
         "stardiv.Toolkit." & strService Then
         Dlg.Controls(intZaehler).Visible = blnSichtbar
      End If
   Next
   Dlg.getControl("lblDatei").Visible = True
End Sub
```

Sie wird an mehreren Stellen, beispielsweise beim Start, aufgerufen:

```
Sub OrgaDialog

   DialogLibraries.LoadLibrary("Organigramm")
   Dlg = CreateUnoDialog(DialogLibraries.Organigramm.dlgOrga)
   Call Unsichtbar("UnoListBoxControl", False)
   Call Unsichtbar("UnoFixedTextControl", False)
   Dlg.Execute()

End Sub
```

Der Benutzer wählt eine Calc-Datei über den Dialog aus oder gibt ihren Namen mit Pfad in das Textfeld ein. Sie wird nun geöffnet. Falls vorher bereits eine andere geöffnet wurde, muss die nicht mehr benötigte Datei geschlossen werden. Von der neuen Datei werden die Tabellenblattnamen in einem Listenfeld angezeigt:

```
Sub TabellenAnzeigen
   Dim strDatei As String
   Dim mArgs(0) As New com.sun.star.beans.PropertyValue
   Dim intZaehler As Integer

   If strDatei <> "" And Dir(strDatei) <> "" Then
      If Not IsNull(objDoc) Then
```

```
            objDoc.Close(True)
        End If
    End If

    Call Unsichtbar("UnoListBoxControl", False)
    Call Unsichtbar("UnoFixedTextControl", False)

    strDatei = ConvertToUrl(Dlg.getControl("FileControl1").Text)
    If Dir(strDatei) <> "" Then
        Dlg.getControl("lstListe").Visible = True
        mArgs(0).Name = "Hidden"
            mArgs(0).Value = True
        objOOo = createUnoService("com.sun.star.frame.Desktop")
        objDoc = objOOo.loadComponentFromURL(strDatei, "_blank", _
            0, mArgs())
        intZaehler = Dlg.getControl("lstListe").ItemCount
            Dlg.getControl("lstListe").removeItems(0, intZaehler)

        For intZaehler = 0 To objDoc.getSheets().Count -1
            Dlg.getControl("lstListe").addItem _
                (objDoc.getSheets().getByIndex(intZaehler).Name, intZaehler)
        Next
    Else
        Dlg.getControl("lstListe").Visible = False
    End If
End Sub
```

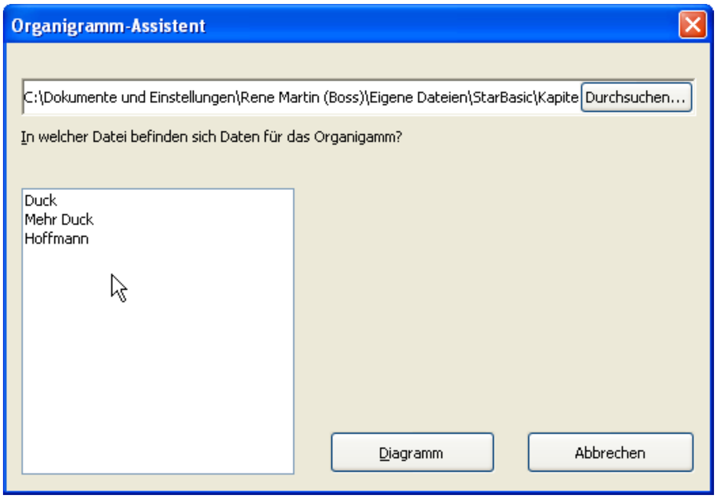


Abbildung 13.196 Die Namen der Tabellenblätter werden angezeigt.

Wird nun eines der Tabellenblätter ausgewählt, dann werden die Spalten angezeigt. Voraussetzung ist allerdings, dass in mindestens fünf Spalten Daten stehen. Die Anzahl der Spalten werden mit dem Cursorobjekt und der Sammlung „Columns" gezählt. Die Kombinationsfelder (eigentlich auch Listenfelder) werden angezeigt und mit den Feldnamen gefüllt.

```
Sub Liste_Click
   Dim intZaehler As Integer
   Dim intSpaltenAnzahl As Integer
   objTabelle = objDoc.getSheets(). _
      getByName(Dlg.getControl("lstListe").selectedItem)
   objZelle = objTabelle.getCellRangebyName("A1")

   objCursor = objTabelle.createCursorByRange(objZelle)
   objCursor.collapseToCurrentRegion()
   intSpaltenAnzahl = objCursor.Columns.Count
   If intSpaltenAnzahl < 5 Then
      MsgBox "Es sind nicht genügend Spalten in der Tabelle " & _
         objTabelle.Name
      Exit Sub
   End If
```

```
   intZaehler = Dlg.getControl("cboID").ItemCount
   Dlg.getControl("cboID").removeItems(0, intZaehler)
   Dlg.getControl("cboName").removeItems(0, intZaehler)
   Dlg.getControl("cboTitel").removeItems(0, intZaehler)
   Dlg.getControl("cboBoss").removeItems(0, intZaehler)
   Dlg.getControl("cboEbene").removeItems(0, intZaehler)
   For intZaehler = 0 To intSpaltenAnzahl - 1
      Dlg.getControl("cboID").addItem _
         (objTabelle.getCellByPosition(intZaehler, 0).String, intZaehler
[...]
   Next

   Dlg.getControl("cboID").SelectItemPos(0, True)
   Dlg.getControl("cboName").SelectItemPos(1, True)
   Dlg.getControl("cboTitel").SelectItemPos(2, True)
   Dlg.getControl("cboBoss").SelectItemPos(3, True)
   Dlg.getControl("cboEbene").SelectItemPos(4, True)
   Call Unsichtbar("UnoListBoxControl", True)
   Call Unsichtbar("UnoFixedTextControl", True)

End Sub
```

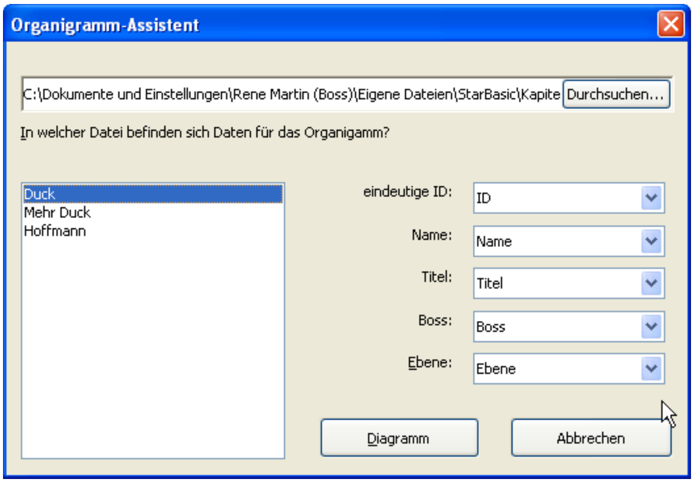


Abbildung 13.197 Die Namen der zugehörigen Spalten werden angezeigt.

Wählt der Benutzer eine Tabelle aus und weist den Informationen „Id“, „Boss“ und „Ebene“ die richtige Spalte zu, dann wird die Tabelle nach den Spalten „Ebene“ und „Boss“ sortiert. Anschließend wird ermittelt, ob die Zeilenanzahl nicht 90 überschreitet (maximal 10 Ebenen × 9 Mitarbeiter) und ob in einer Ebene nicht mehr als neun Mitarbeiter stehen. Schließlich wird geprüft, ob in den drei ID-Spalten nur Zahlen stehen. Prinzipiell könnte man auch Texte verwenden, allerdings muss dann der Inhalt der Calc-Zelle mit der Eigenschaft `String` und nicht mit `Value` ausgelesen werden. Auf eine Plausibilitätsprüfung, beispielsweise ob jede Ebene gefüllt ist, ob Mitarbeiter nur Chefs der nächsthöheren Ebene haben und so weiter, wurde verzichtet. Dann werden die Namen, Nummern der Vorgesetzten und eigene ID-Nummern ausgelesen und in einem dreidimensionalen Array gespeichert. Ebenso wird die Anzahl der Mitarbeiter pro Ebene in einem Datenfeld abgelegt:

```
Sub Neues_Diagramm
[...]
   For intZaehler = 1 To intZeilenAnzahl - 1

       If objTabelle.getCellByPosition(intEbene, intZaehler).Value = _
        objTabelle.getCellByPosition(intEbene, intZaehler - 1).Value Then
           intMitarbeiterZaehler = intMitarbeiterZaehler + 1
       Else
           intMitarbeiterZaehler = 1
```

```
            intEbenenZaehler = intEbenenZaehler + 1
        End If

        strEbenennummer(intEbenenZaehler, intMitarbeiterZaehler, 1) = _
           objTabelle.getCellByPosition(intName, intZaehler).String & _
           Chr(10) & _
           objTabelle.getCellByPosition(intTitel, intZaehler).String
        'Inhalt

        strEbenennummer(intEbenenZaehler, intMitarbeiterZaehler, 2) = _
           objTabelle.getCellByPosition(intBossNr, intZaehler).Value
        'Boss

        strEbenennummer(intEbenenZaehler, intMitarbeiterZaehler, 3) = _
           objTabelle.getCellByPosition(intID, intZaehler).Value
        'Eigene ID

        intMitarbeiterAnzahl(objTabelle.getCellByPosition(intEbene, _
           intZaehler).Value) = intMitarbeiterZaehler
    Next intZaehler
[...]
```

Nun kann eine leere Zeichnung erstellt werden:

```
    strUrl = "private:factory/simpress"
    objDraw = StarDesktop.loadComponentFromURL(strUrl, "_blank", 0, _
       docArgs())
    objSeite = objDraw.getDrawPages().getByIndex(0)
    objSeite.Orientation = com.sun.star.view.PaperOrientation.LANDSCAPE
```

Die Breite der Rechtecke, der linke obere Eckpunkt und die Abstände werden festgelegt:

```
    dblBreite = 2500
    dblHoehe = 1000
    dblMassY = 1000
    dblMassX = 13000 ' hier ist die Mitte
```

```
dblAbstand = 500
```

Zwei ineinander geschachtelte Schleifen durchlaufen die Mitarbeiter einer Ebene und die Anzahl der Ebenen. Mit ein wenig Arithmetik werden die Rechtecke positioniert. Da eine gerade Anzahl Mitarbeiter versetzt zu einer ungerade Anazahl steht, wird mit Mod 2 gearbeitet – damit kann gerade und ungerade leicht in einer Rechenzeile abgefangen werden:

```
For intZaehler = 1 To intEbenenAnzahl
   For intEbenenZaehler = 1 To intMitarbeiterAnzahl(intZaehler)
      Punkt.X = dblMassX + (intMitarbeiterAnzahl(intZaehler) Mod 2 - _
      1) * dblBreite / 2 - Fix(intMitarbeiterAnzahl(intZaehler) / 2 _
      - 1) * (dblBreite + dblAbstand) + _
      (intEbenenZaehler - 1) * (dblBreite + 500)
      Punkt.Y = (intZaehler * 2 - 1) * dblMassY
      Groesse.Width = dblBreite
      Groesse.Height = dblHoehe
      objRechteck = _
         objDoc.createInstance("com.sun.star.drawing.RectangleShape")
      objRechteck.Size = Groesse
      objRechteck.Position = Punkt
      objSeite.add(objRechteck)
```

Das Rechteck wird weiß eingefärbt und mit dem Text des Datenfeldes beschriftet:

```
      objRechteck.FillColor = RGB(255, 255, 255)
      objRechteck.String = _
         strEbenennummer(intZaehler, intEbenenZaehler, 1)
```

Der Text wird an das Objekt angepasst und mit Abständen zu den vier Rändern versehen. Dem Rechteck wird als Name eine Kombination aus eigener ID und Boss-ID zugewiesen:

```
[...]
objRechteck.Name = strEbenennummer(intZaehler, intEbenenZaehler, 2) & _
   "/" & strEbenennummer(intZaehler, intEbenenZaehler, 3)
[...]
```

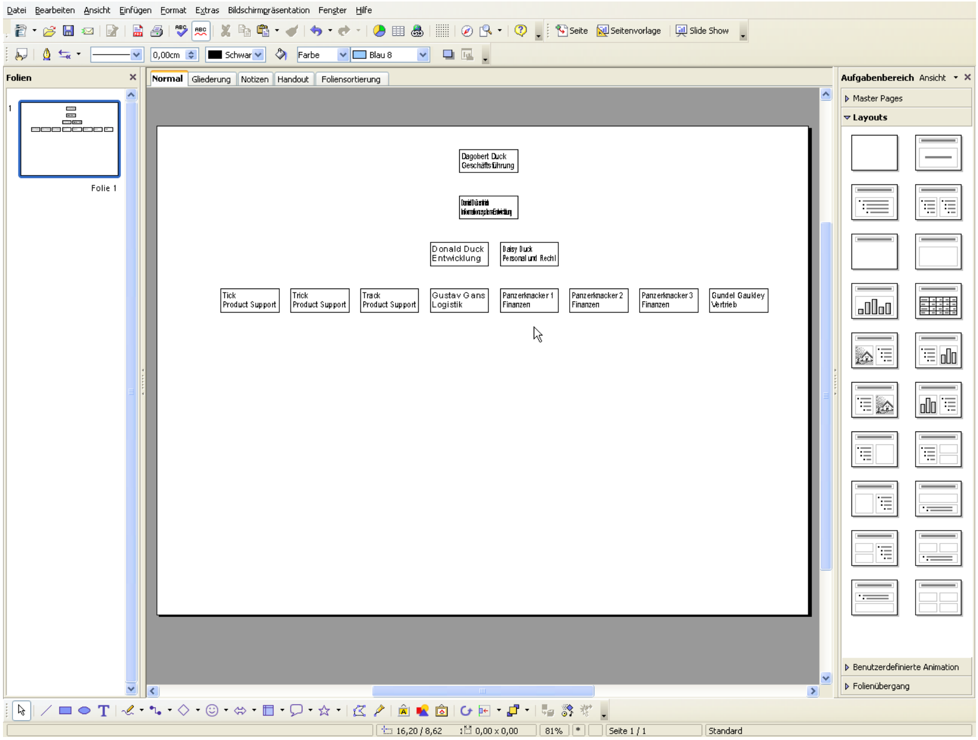


Abbildung 13.198 Das fertige Diagramm

Anschließend werden sämtliche Objekte der Seite durchlaufen. Für jedes Objekt, das nicht Chef ist, wird ein Verbinder erzeugt, der den Boss mit seinem Mitarbeiter verknüpft:

```
[...]
   objVerbinder = _
      objDoc.createInstance("com.sun.star.drawing.ConnectorShape")
      objRechteck = objSeite.getByIndex(intEbenenZaehler)
      objSeite.add(objVerbinder)
      objVerbinder.StartShape = objRechteck
      objVerbinder.StartGluePointIndex = 2
      objRechteck = objSeite.getByIndex(intZaehler)
      objVerbinder.EndShape = objRechteck
      objVerbinder.EndGluePointIndex = 0
      Exit For
```

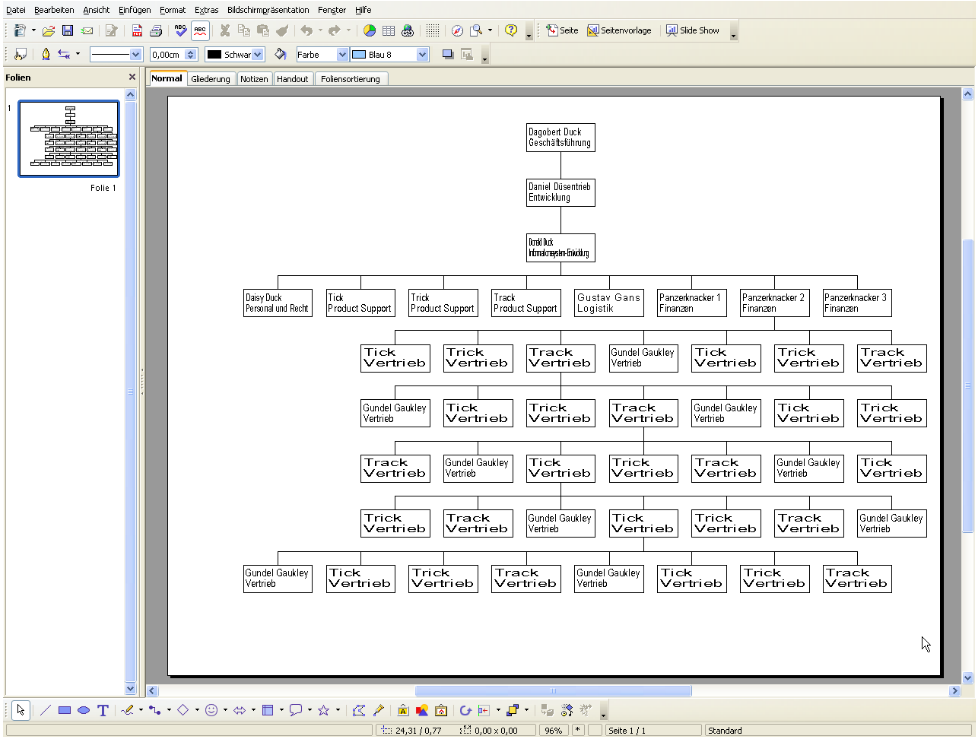


Abbildung 13.199 Ein Diagramm mit Verbinder-Linien

Zum Schluss werden die Calc-Datei und der Dialog geschlossen:

```
[...]
    objDoc.Close(True)
    Dlg.endExecute
```

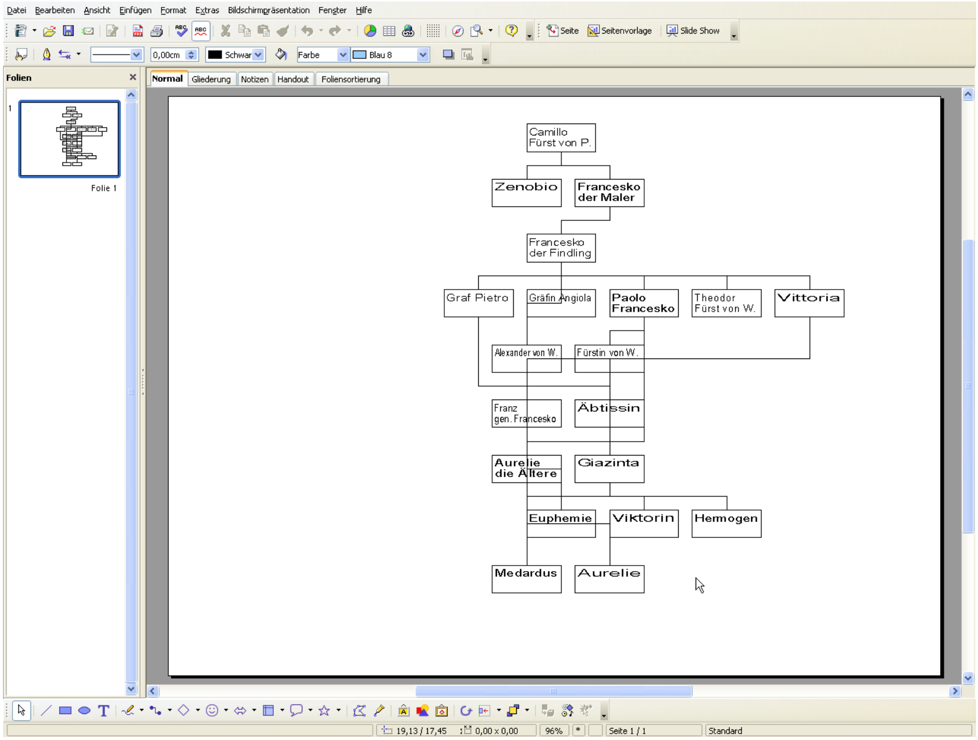


Abbildung 13.200 Diese (nicht sehr schöne) Familiengenealogie muss noch nachbearbeitet werden.

Ein Organigramm-Assistent

Ähnlich wie in Kapitel 11 werden hier zwei Briefvorlagen vorgestellt, mit der schnell Briefe erzeugt werden können. In diesem Kapitel geht es allerdings nicht um das Generieren von Briefen, sondern um den Zugriff von Briefen auf eine Datenbank. Der Benutzer selektiert den richtigen Eintrag, und das Programm fügt die zugehörigen Informationen an die entsprechenden Stellen ein.

14Ein Briefassistent

Damit die beiden Beispiele funktionieren, muss eine Datenbankverknüpfung zu einer bestehenden Datenbank angelegt werden. In unserem Beispiel besteht die Datenbank „Verlage“ lediglich aus einer Tabelle „tblVerlage“.

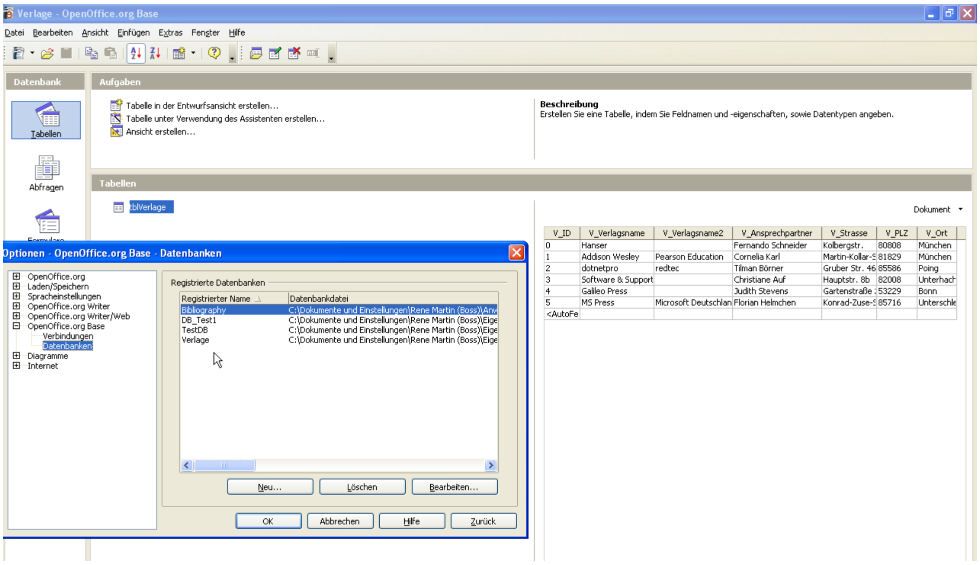


Abbildung 14.201 Die Datenbank wird erstellt und registriert.

In einer Briefvorlage wird ein Dialog mit Listenfeld, einem Label und zwei Schaltflächen erstellt:

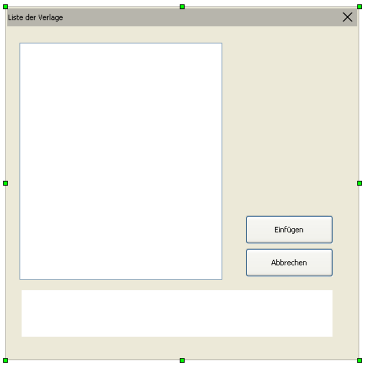


Abbildung 14.202 Der Dialog, der mit den Daten gefüllt wird

14.1 Zugriff auf die Datenbank

Wie in Kapitel 8 beschrieben, kann auf die registrierte Datenbank zugegriffen werden:

```
Dim Dlg As Object
Dim strAnsprechpartner() As String

Sub Dialog_Start
   Dim objDatabaseContext As Object
   Dim objDataSource As Object
   Dim objConnection As Object
   Dim objInteractionHandler as Object
   Dim objStatement As Object
   Dim objResultSet As Object
   Dim strListe As String
   Dim i As Integer

   DialogLibraries.LoadLibrary("DBZugriff")
   Dlg = CreateUnoDialog(DialogLibraries.DBZugriff.dlgVerlage)

   objDatabaseContext = _
      createUnoService("com.sun.star.sdb.DatabaseContext")
   objDataSource = objDatabaseContext.getByName("Verlage")
   If Not objDataSource.IsPasswordRequired Then
      objConnection = objDataSource.GetConnection("","")
   Else
      objInteractionHandler = _
         createUnoService("com.sun.star.sdb.InteractionHandler")
      objConnection = _
         objDataSource.ConnectWithCompletion(objInteractionHandler)
   End If
   objStatement = objConnection.createStatement()
```

```
  objResultSet = _
     objStatement.executeQuery("SELECT * FROM ""tblVerlage""")
```

Das Resultset wird durchlaufen, der Inhalt der zweiten Spalte wird im Listenfeld angezeigt, und die Daten der vierten Spalte werden in einem Datenfeld gespeichert.

Beachten Sie, dass die Zählung der Spalten beim Datenbankzugriff anders als bei allen anderen Objekten bei 1 beginnt!

```
  i = 0

  ReDim Preserve strAnsprechpartner(i)
  If Not IsNull(objResultSet) Then
     While objResultSet.next
        Dlg.getControl("lstListe").addItem(objResultSet.getString(2), i)
        strAnsprechpartner(i) = objResultSet.getString(4)
        i = i + 1
        ReDim Preserve strAnsprechpartner(i)
     Wend
  End If
```

Anschließend wird die Verbindung zur Datenbank getrennt. Man könnte alternativ alle Daten in Datenfelder einlesen und trennen oder die Verbindung stehen lassen, bis der Benutzer den Dialog schließt. Diese Variante hat den Vorteil, dass er bei geöffneter Maske Kaffee trinken gehen kann. Kehrt er zu seinem Rechner zurück, wird die Verbindung erneut aufgebaut.

```
  objResultSet.Close()
  objStatement.Close()
  objConnection.Close()

  Dlg.Execute()
End Sub
```

Im Label wird der zugehörige Ansprechpartner angezeigt, wenn ein Verlagsname ausgewählt wurde:

```
Sub Liste_Click
```

```
    Dim intPos As Integer
    intPos = Dlg.getControl("lstListe").selectedItemPos
    Dlg.getControl("lblInfo").setText(strAnsprechpartner(intPos))

End Sub
```

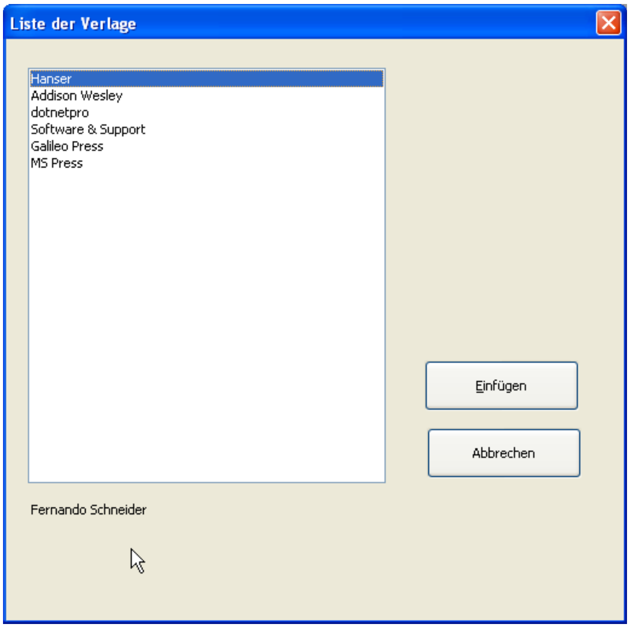


Abbildung 14.203 Die Verlagsnamen werden aufgelistet, der zugehörige Lektor angezeigt.

Klickt der Benutzer auf die Schaltfläche „Einfügen“, wird erneut eine Datenbankverbindung erzeugt, und die Daten werden eingetragen. Damit der Code flexibel bleibt, durchläuft ein Zähler sämtliche Spaltennamen und überprüft, ob es im Dokument entsprechende Textmarken gibt. Falls ja, wird ein Cursorobjekt an der entsprechenden Stelle erzeugt, wohin anschließend der Text geschrieben wird:

```
Sub cmdEintragen_Click

    Dim objDatabaseContext As Object
    Dim objDataSource As Object
    Dim objConnection As Object
```

```
Dim objInteractionHandler as Object
Dim objStatement As Object
Dim objResultSet As Object
Dim strSpalte As String
Dim strInhalt As String
Dim i As Integer
Dim objDatei As Object
Dim objTextmarken As Object
Dim objTextmarke As Object
Dim objCursor As Object

objDatei = ThisComponent
objDatabaseContext = _
   createUnoService("com.sun.star.sdb.DatabaseContext")
objDataSource = objDatabaseContext.getByName("Verlage")
If Not objDataSource.IsPasswordRequired Then
   objConnection = objDataSource.GetConnection("","")
Else
   objInteractionHandler = _
      createUnoService("com.sun.star.sdb.InteractionHandler")
   objConnection = _
      objDataSource.ConnectWithCompletion(objInteractionHandler)
End If
objStatement = objConnection.createStatement()

objResultSet = objStatement.executeQuery("SELECT * FROM " & _
   """tblVerlage"" WHERE ""V_Verlagsname"" = '" & _
   Dlg.getControl("lstListe").selectedItem & "'")

If Not IsNull(objResultSet) Then
   While objResultSet.next
      For i = 0 To objResultSet.Columns.Count - 1
```

```
        strSpalte = Right(objResultSet.Columns(i).Name, _
           Len(objResultSet.Columns(i).Name) - 2)

           objTextmarken = objDatei.getBookmarks()
            If objTextmarken.hasByName(strSpalte) = True Then
               objTextmarke = objTextmarken.getByName(strSpalte)
               objCursor = _
                  objDatei.Text. _
                     createTextCursorByRange(objTextmarke.Anchor)
               objCursor.String = objResultSet.getString(1)

             End If
      Next
     Wend
 End If
 objResultSet.Close()
 objStatement.Close()
 objConnection.Close()
 End Sub
```

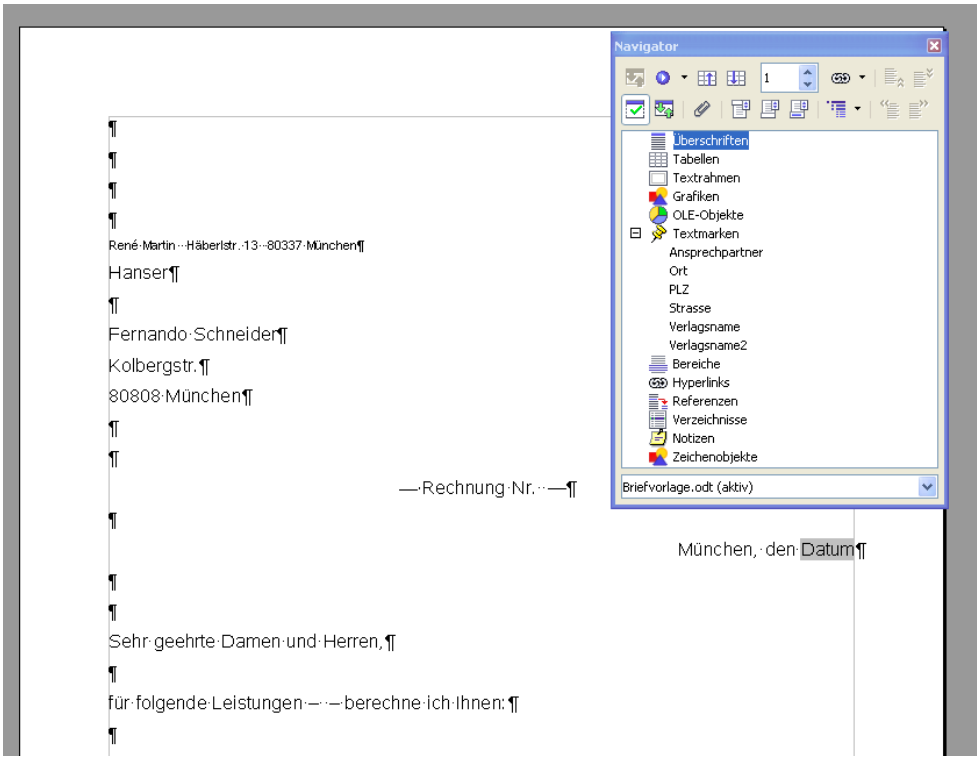


Abbildung 14.204 Die Daten werden in das Writer-Dokument geschrieben. Der Navigator zeigt die Liste der Textmarken an.

Index

- 29
' 8
#Name?......................201
$ 35
%20............................167
& 32
* 29
.NET...........................333
/ 29
: 7
^ 29
_ 6
+ 29, 32
< 39
<=..............................39
<>..............................39
= 39
> 39
>=..............................39

A

Abbrechen-Button.......103
Abfragen......................420
abs................................30
Absatz.................272, 275
absoluter Bezug...........195
Achsen.........................265
ActiveFrame................169
addItem........114, 119, 120
AdjustBlue..................389
AdjustContrast.............389
AdjustGreen................389
AdjustLuminance........389
Access.........................418
AdjustRed...................389
Ähnlichkeitssuche.......331
Alphabeth....................290
Alignment...................309
AllowAnimations........408
AnchorFrame..............354
AnchorType.........337, 354
And.........................39, 83
Anführungszeichen.......34
Animation...................408
Animationseffekte.......413
Annotation...........225, 360
ANSI-Code...................32
Anzahl der Tabellenblätter 183
API.............................144
API-Referenz..............148
Application Programming Interface.....................144
Array.....................36, 151
Arrays..........................24
Asc................................37
asiatische Texte...........297
Assistent für Organigramme 484
AsTemplate.................171
Aufhebungsvertrag......473
ausfüllen......................227
Ausrichtung.................303
Autor...........................360

B

BackColor...................313
Background.................373
Balkendiagramm.........262
Balkengrafik................262
Basic............................144
Bedingte Formatierung223
Bedingungsschleife.......62
beenden.......................177
Beep..............................16
benutzerdefinierte Animationen...........................413
Beobachter.....................93
Beobachtungsfenster.......4
Beschriftungsfelder.....105
Bibliography................360
Bibliothek.......................2

Bibliotheken................100
Bildschirmpräsentations-
einstellungen...........408
Bitmap.........................394
Bitmapmuster......391, 394
Bitweiser Vergleich.......83
Blätter kopieren...........183
Blätter löschen.............183
Blattschutz...................189
Bodenfläche.................268
Bookmarks..................351
Boolean...................21, 23
BottomBorder......305, 316
BottomBorderDistance316
BottomMargin.............315
BreakType...................305
Breite...........................378
Buchstabendreher........289

C

C++..........................2, 144
Calc.............................179
Call................................59
CamelCasing...................4
Case..............................42
Case Is...........................42
CBool............................24
CDate............................24
CDateFromIso.............131
CDateFromISO.............24
CDateToIso.................131
CDateToISO.................24
CDbl.............................24
Cell.Type.....................200
CellBackColor.............208
CellDeleteMode..........204
CellProtection..............214
Change.........................409
Chapter........................360
CharacterCount...........360
CharBackColor............296
CharBackTransparent..296
CharColor....................296
CharEscapement..........296
CharEscapementHeight296
CharFontName............295
CharFontStyleName....296
CharHeight..................295
CharShadowed............296
CharStyleName...........296
Charts..........................260
ChartShape..................377
CharUnderline.............295
CharWeight.................295
ChDir............................82
ChDrive.........................82
Choose..........................46
Chr................................37
Chr$(10)................33, 279
Chr$(13)................33, 279
Chr$(9)..................33, 279
CInt...............................24
CircleEndAngle...........379
CircleKind...................379
CircleStartAngle..........379
clearContents..............205
CLng.............................24
Close......................86, 177
ClosedBezierShape.....376
collapseToEnd.....287, 288
collapseToStart....287, 288
Collatz’sche Problem....66
Columns.............190, 193
com.sun.star..style. DropCap-
Format.....................305
com.sun.star.awt. FontFamily
................................295
com.sun.star.awt. FontUnder-
line..........206, 263, 295
com.sun.star.awt. FontWeight
...............206, 263, 295
com.sun.star.awt.
Gradient..................392
com.sun.star.awt.
Point 288, 374, 377, 387
com.sun.star.awt.
Size................176, 377
com.sun.star.chart.
ChartArea................268
com.sun.star.container.
XNameContainer....306
com.sun.star.DataPilot-Fiel-
dOrientation............257
com.sun.star.document.
OfficeDocument.....166
com.sun.star.drawing.
BitmapMode...........394
com.sun.star.drawing.
CircleKind...............379
com.sun.star.drawing.
ColorMode..............389
com.sun.star.drawing.
ConnectorShape404, 405
com.sun.star.drawing.
ConnectorType.......404
com.sun.star.drawing.
DrawingDocument..371
com.sun.star.drawing.
EllipseShape...........379
com.sun.star.drawing.Fill-
Properties 264, 377, 391
com.sun.star.drawing.
FillStyle..................392

com.sun.star.drawing.
GluePoint2..............405
com.sun.star.drawing.
GraphicObjectShape389
com.sun.star.drawing.
GroupShape............399
com.sun.star.drawing.
Hatch.......................393
com.sun.star.drawing.
LineProperties. 377, 382
com.sun.star.drawing.
LineShape...............382
com.sun.star.drawing.
LineStyle.................264
com.sun.star.drawing.
PolyPolygonShape 375, 387
com.sun.star.drawing.
RectangleShape.......377
com.sun.star.drawing.
RotationDescriptor..404
com.sun.star.drawing.
ShadowProperties. 377, 383, 398
com.sun.star.drawing.
Shape.......................374
com.sun.star.drawing.
ShapeCollection......398
com.sun.star.drawing.
Text................377, 382
com.sun.star.drawing.Text-
HorizontalAdjust.....396
com.sun.star.drawing.-
TextProperties.........396
com.sun.star.drawing.-
TextVerticalAdjust. 397
com.sun.star.frame.
Desktop...................166
com.sun.star.frame.
XComponentLoader166
com.sun.star.frame.
XStorable................172
com.sun.star.
GeneralFunction.....258
com.sun.star.presentation.An-
imationEffect..........413
com.sun.star.presenta-
tion.ClickAction......416
com.sun.star.presentation.-
FadeEffect..............410
com.sun.star.presentation.P-
resentationDocument371
com.sun.star.sdb.
DataSource.............419
com.sun.star.sheet.Cell-
Flags.ANNOTATION205
com.sun.star.sheet.
FillDirection............228
com.sun.star.sheet.
FillMode.................228
com.sun.star.sheet.Filter-
Connection.AND....259
com.sun.star.sheet.Filter-
Connection.OR.......259
com.sun.star.sheet.Filter-
Operator.................258
com.sun.star.sheet.
Spreadsheet.............184
com.sun.star.sheet.
TablePageStyle.......248
com.sun.star.sheet.
XPrintAreas............176
com.sun.star.sheet.
XRangeMovement..202
com.sun.star.style.
BreakType.......305, 341
com.sun.star.style.
CharacterProperties206, 295, 377, 383
com.sun.star.style.
GraphicLocation305, 319, 342
com.sun.star.style.
LineSpacing............303
com.sun.star.style.
NumberingRule.......397
com.sun.star.style.
NumeringType........364
com.sun.star.style.
PageProperties242, 313, 315, 317, 319
com.sun.star.style.
ParagraphAdjust.....304
com.sun.star.style.
ParagraphProperties303, 377, 383
com.sun.star.style.
TabAlign.................309
com.sun.star.style.
TabStop...................304
com.sun.star.table.
BorderLine210, 305, 315, 318, 341
com.sun.star.table.Cell-
ContentType...199, 347
com.sun.star.table.Cell-
HoriJustify..............212
com.sun.star.table.Cell-
Orientation..............212
com.sun.star.table.Cell-
Properties................208
com.sun.star.table.Cell-
RangeAddress.........202

com.sun.star.table.CellVertJustify...............212
com.sun.star.table.ShadowFormat.....305, 316, 342
com.sun.star.table.ShadowLocation.....305, 342
com.sun.star.table.TableBorder............210
com.sun.star.table.TableColumn..........191
com.sun.star.table.TableColumns.........190
com.sun.star.table.TableSortField........251
com.sun.star.text.ControlCharacter.....281
com.sun.star.text.HoriOrientation.......341
com.sun.star.text.Paragraph272, 273, 275, 295, 303
com.sun.star.text.ParagraphVertAlign 304
com.sun.star.text.TextContent..........................337
com.sun.star.text.TextContentAnchorType337, 354, 356
com.sun.star.text.TextCursor................................328
com.sun.star.text.TextField.Annotation274, 369
com.sun.star.text.TextField.DateTime.......274
com.sun.star.text.TextField.FileName.......274
com.sun.star.text.TextField.PageNumber. .366
com.sun.star.text.TextFrame................................358
com.sun.star.text.TextPortion................................295
com.sun.star.text.TextTable................272, 273, 347
com.sun.star.text.UserDataPart...........362
com.sun.star.text.WrapTextMode.......337
com.sun.star.Text.WrapTextMode.......355
com.sun.star.text.XText......................319
com.sun.star.text.XTextRange............278
com.sun.star.util.CellProtection.........214
com.sun.star.util.SearchDescriptor.....329
com.sun.star.view.PaperOrientation.....373
com.sun.star.view.XPrintable...............172
CombinedCharacters...360
Command....................421
Concatenation................32
ConditionalFormat......223
ConditionalText...........360
Connectorshape...376, 404
Const.............................27
Controls.......................102
ControlShape...............376
ConvertFromUrl....38, 167
ConvertToUrl 38, 131, 167
copyByName...............186
CopyCount..................175
copyRange...................204
CornerRadius...............377
cos...........................30, 70
createCursor................229
createCursorByRange. 229
createDataPilotDescriptor257
createEnumeration157, 273
createInstance..............147
createSearchDescriptor329
createTextCursor.........278
createTextCursorByRange................................279
CreateUnoDialog.........145
CreateUnoListener......145
createUnoService 145, 166
CreateUnoStruct..........145
CreateUnoValue..........145
CSng.............................24
CStr...............................24
CurDir...........................82
Currency..................22, 23
Cursor.................287, 288
Cursor ausdehnen........283
Cursor-Objekt..............278
CustomShow...............408

D

Database..............360, 418
DatabaseContext.........418
Date.......18, 22, 23, 46, 47
Dateiauswahl...............131
Dateizugriff...................81
Daten trennen..............233
Datenfeld...............24, 151
Datenpilot....................255
Datenquelle.................418
Datenreihen ausfüllen..227
DateSerial................45, 47
DateValue..........45, 46, 47

DATUM........................77
Datums- und Uhrzeitformate
................................214
Datumsfelder...............130
Day....................45, 46, 47
DBG_methods.............149
DBG_properties..........149
DBG_supportedInterfaces149
Debug-Eigenschaften. .149
DecimalChar...............310
Declare..........................88
Diagramm............260, 384
Diagrammhintergrund. 268
Diagrammwände.........268
Dialog........................8, 99
DialogLibraries...........145
Diawechsel..................409
Dim................................25
DIN A5........................314
Dir.................................82
Diskriminante................72
dispose.........................177
dll-Dateien.....................88
Do ... Loop....................62
Double.....................22, 23
Draw............................371
DrawPages...................371
drehen..........................404
Drehfeld.......................130
DropCapCharStyleName306
DropDownLineCount..120
drucken........................174
Drucker........................176
Duration.......................409
dynamisches Datenfeldes25

E

E 76
Editable.......................120
Effect...........................410
Effekt...........................415
Eigenschaften..............148
Eingabemaske...............99
Einzug.........................303
Ellipse..........................379
Ellipsenabschnitt.........379
EllipseShape................376
Else...............................40
Enable..............................
102, 106, 112, 119, 120
End Sub...........................4
endExecute..................102
Endlosschleife…62, 64, 94
Endquersumme.............77
Entgeltbrief.................458
Entwicklungsumgebung 80
EqualUNOObjects.......145
Eqv................................39
Ereignisse......................80
Ersetzen von Textteilen332
Escape-Kodierung.......167
execute.........................102
Exit For..........................62
Exit Sub.........................97
exp.................................30
Expand.........................282
explizite Deklaration.....20
ExtendedUser..............362

F

Fakultät..........................67
Farbmodus...................389
Farbverläufe................391
Fehler.............................89
Felder...........................337
Fermat’sche These........66
fett...............................294
Fettdruck.....................285
Fibonacci.......................69
file:///...........................167
FileCopy........................82
FileDateTime.................82
FileLen..........................82
FileName.....................362
FillChar.......................310
fillSeries..............227, 228
FillStyle.......................392
FillTransparenceGradient-
Name.......................392
FilterOptions...............171
FindObject..................145
FindObjectProperty.....145
fix.....................................
..................................30
Flächenattribute...........391
Fließtext.......................353
Folienübergänge..........409
FooterIsOn..................318
FooterLeftMargin........318
FooterRightMargin......318
FooterText...................319
FooterTextLeft............319
FooterTextRight..........319
For ... Next....................61
For Input........................86
For Output.....................87
Format...........................49
formatieren..................294
formatierte Felder........130
Formatvorlagen…321, 322
Formeln.......................197
Formula.......................197
FormulaLocal..............201
Formular........................99

FormulaResultType.....201
Fortschrittsbalken........129
FrameHeightAbsolute. 354
FrameHeightPercent....354
FrameIsAutomaticHeight354
Frames.........................337
FrameWidthAbsolute. .355
FrameWidthPercent.....355
Füllmodus....................392
Function.........................53
Funktionen...............53, 71
Fußzeile.......................365

G

Gamma........................389
Geburtstag.....................45
Geschäftsjahr.................43
GetAttr...........................81
getBookmarks.............351
getCellByName...........346
getCellByPosition194, 346
getCellFormatRanges. .219
getCellNames..............346
getCellRangeByName. 194
getColumns.................344
getConnection.............423
getCurrentSelection.....303
getDate........................130
GetDefaultContext......145
getDrawPages..............371
getDropDownLineCount120
getElementNames.......156
getEnd.........................288
getError.......................201
GetExpression.............362
getFormula..................201
getItem........................119
getItemCount.......119, 120
getLocation.................173
getPage........................288
getPosition..................288
GetProcessServiceManager
................................145
getRangeAddress.195, 196
GetReference..............362
getRows......................344
getSelectedItem...........119
getSelectedItemPos.....119
getSelectedItems. 117, 119
getSelectedItemsPos118, 119
getSelectedText...........120
getSheets.....................180
getStart........................288
getString......277, 287, 288
getText.........................288
getTextTables..............339
getTime.......................131
getTitle........................102
getType........................199
ggT................................76
Gleichheitszeichen........18
GlobalScope................145
GluePoint2..................405
goDown.......................288
GOLDENERSCHNITT 76
goLeft.................282, 288
goRight........280, 282, 288
GoTo.............................71
gotoEnd.......282, 285, 288
gotoEndOfLine............288
gotoEndOfParagraph...282
gotoEndOfSentence.....282
gotoEndOfWord..........282
gotoNextParagraph......282
gotoNextSentence.......282
gotoNextWord.............282
gotoPreviousParagraph282
gotoPreviousSentence. 282
gotoPreviousWord.......282
gotoRange...........282, 288
gotoStart......282, 285, 288
gotoStartOfLine...........288
gotoStartOfParagraph..282
gotoStartOfSentence...282
gotoStartOfWord.........282
goUp............................288
GRADIENT................392
Grafik.................337, 389
Grafische Kontrollfelder127
GraphicColorMode.....389
GraphicObjectCount...363
GraphicObjectShape...376
GraphicURL................389
Groß- und Kleinschreibung
............................7, 158
group...........................398
GroupShape.................376
gruppieren...................398
Gültigkeit....................221

H

Haltepunkte...................92
HandoutShape.............377
HARD_HYPHEN.......281
HARD_SPACE...........281
hasLocation.................173
hasMoreElements........157
HasUnoInterfaces........145
HATCH.......................393
HeaderIsOn.................318
HeaderLeftMargin.......318
HeaderRightMargin.....318
HeaderText..................319
HeaderTextLeft...........319

HeaderTextRight.........319
Height..........191, 313, 377
Helligkeit.....................390
HEUTE().......................76
HiddenParagraph.........363
HiddenText..................363
Hintergrundfarbe. 295, 303
Höhe............................378
Hour.........................46, 47
HTML.................275, 277
Hurenkinderregelung...304
HyperLinkName..........296
HyperLinkTarget.........296
HyperLinkURL...........296
Hypothenuse..................78

I

IDE..................................4
If 40
IIf..................................42
Imagecontrols..............127
Imp................................39
ImplementationName........
................102, 106, 112
Implizite Deklaration....20
Impress........................371
Index............................432
Informationsabfragen....60
Inkrement....................227
Input............................363
InputBox........................16
InputUser.....................363
insertByIndex..............193
insertByName..............184
insertCell.....................202
insertControlCharacter 279
insertTextContent........283
InStr.........................34, 37
int..................................30
Integer.....................21, 23
Interface.......................147
Internet-Standard RFC 1738
........................131, 167
Intervall.......................107
IsArray...........................60
isAtEndOfLine............288
isAtStartOfLine...........288
IsAutomatic.................409
isCollapsed..........287, 288
IsDate............................60
isEditable.....................120
IsEmpty.................60, 152
isEndOfParagraph.......287
isEndOfWord.............287
IsHidden......................214
IsLandscape.........243, 313
IsLivePresentation.......409
IsLocked......................214
isModified...................173
IsMouseVisible...........409
isMutipleMode....117, 119
IsNull............................60
IsNumeric......................60
IsObject.........................61
isProtected...................189
isReadonly...................173
isStartOfParagraph......287
isStartOfWord.............287
IsSyncHeightToWidth.355
IsSyncWidthToHeight.355
IsTransitionOnClick....409
IsUnoStruct...........61, 145
isVisible.......................288
IsVisible......................181
ItemCount....114, 119, 120
Items....................119, 120

J

Java..................2, 144, 334
JavaScript......................76
JETZT().........................76
Join..........................36, 37
JumpEdit.....................363
jumpToEndOfPage......288
jumpToFirstPage.........288
jumpToLastPage.........288
jumpToNextPage.........288
jumpToPage................288
jumpToPreviousPage. .288
jumpToStartOfPage.....288

K

Kalender......................130
Kathete..........................78
Kennwort.....................190
kgV...............................76
Kill.................................82
Klebepunkt..................405
Kombinationsfeldern...121
kombinieren.................403
Kommentare....................8
Kommentarfarben...........9
Konjunktoren.................38
Konstanten.........8, 27, 158
Kontrast...............389, 390
Kontrollelement...........133
Kontrollfelder..............102
Kontrollkästchen.........110
Kontrollstrukturen.........38
Koordinate...................378
Kopf- und Fußzeilen.........
................246, 312, 317
Kreis............................379
kursiv...........................294

L

Label....................106, 112
Laufwerk........................84
Lcase..............................37
Left..........................34, 36
LeftBorder...........305, 315
LeftBorderDistance.....316
LeftMargin..................315
Legende.......................264
Len.................................37
Leuchtkraft..................389
LINE_BREAK............281
LineColor....................395
LineDash.....................395
LineEnd.......................395
LineStart......................395
LineStyle.....................395
Linien..........337, 347, 382
Linienattribute.............395
Linienmuster...............393
LinieShape...................376
Listenfelder.........114, 121
loadComponentFromURL
........................155, 166
loadCompontentFromURL
................................167
loadFromURL.............173
LocalSetting.Country. .216
LocalSetting.Language215
log.................................30
logische Operatoren......39
Long........................21, 23
Ltrim........................34, 37
Luminanz.....................389

M

ma................................107
Macro..........................363
makeVisible................120
Map AppFont..............107
Marke............................71
Markierfelder..............110
Markierung..................292
Maskiertes Feld...........131
Max. Wert...................107
MeasureShape.............376
Mehrfachauswahl........121
Mehrfachselektion............
................121, 233, 292
Mehrseitige Dialoge....123
Menüpunkte...................80
Merge..........................220
Methoden....................148
Mid..........................35, 37
Min. Wert....................107
Minute.....................46, 47
MkDir............................82
Model.Date.................130
Model.Enabled.................
102, 106, 112, 119, 120
Model.Height......106, 112
Model.Label................106
Model.PositionX..............
................106, 112, 122
Model.PositionY. 106, 112
Model.TabIndex. .106, 112
Model.Tabstop....106, 112
Model.Time.................131
Model.Width.......106, 112
Models Properties........133
Module..........................79
MONAT........................77
Month................45, 46, 47
movebyName..............186
MsgBox.........................11
MultipleMode......117, 119

N

Nachkommastellen......107
Name............82, 101, 106
nextElement................157
Not................................39
NotesShape.................377
Notiz............225, 360, 367
Now...............................47
nSearchFlags...............170
NumberingStyleName. 306

O

objectwitharrow...........406
objectwithshadow........406
Objekte........................144
öffnen..........................166
OLE2Shape.................376
On Error GoTo..............89
On Error Resume Next..89
Open.............................86
OpenBezierShape........376
OptimalHeight.............191
OptimalWidth..............191
Option Explicit..............19
Optionsfelder...............108
Or39
Ordner...........................84
Orientation...................373
Ostersonntag..................31
OutlinerShape..............377

P

PageCount...................363
PageNumber................363
PageShape...................376

PaperFormat................176
PaperOrientation.........175
PaperSize.....................176
Papierformat................312
ParaAdjust...................304
ParaBackColor............305
ParaBackTransparent. .305
ParaBottomMargin......303
ParaFirstLineIndent.....304
PARAGRAPH_BREAK281
ParagraphCount...........363
ParaIsAutoFirstLineIndent
................................304
ParaIsHyphenation......304
ParaLeftMargin...........303
ParaLineSpacing.........303
ParaOrphans................304
ParaRightMargin.........303
ParaShadowFormat.....305
ParaStyleName............306
ParaTabStops...............304
ParaTopMargin...........303
ParaUserDefinedAttributes
................................306
ParaVertAlignment.....304
ParaWidows................304
Pearl............................333
Pfeilspitze............384, 407
PHP.............................333
PI(),...............................76
PluginShape.................376
Polygon.......................387
PolyLineShape............376
PolyPolygonBezierShape376
PolyPolygonShape......376
Position........294, 309, 377
Präsentation.................371
Presentation.................408
Preserve.........................26
Primzahl........................74
print.......................15, 174
PrinterPaperTray.........313
Private...........................25
Progessbar...................129
ProgressValueMax......129
ProgressValueMin.......129
Properties....................152
protect..........................189
Prozent...........................30
Public.............................25
PutCell........................198
Python.........................144

Q

Quadrat........................377
quadratische Gleichung. 72
quer.............................314
Querydefinition...........420
Quersumme...................77
Querverweis........337, 362
queryKey.....................214
Quickinfo.......................93

R

Radiobuttons...............108
Rahmen.......................108
randomize......................30
READ_ONLY.............428
Rechnungsnummer........85
Rechteck......................377
rechtwinkliges Dreieck..78
RectangleShape...........377
ReDim...........................25
ReferencePageGet.......363
ReferencePageSet........364
reguläre Ausdrücke.....333
Rekursion......................67
RelativeHeight.............355
relativer Bezug............195
RelativeWidth.............355
Rem.................................8
removeByIndex...........193
removeByName...........183
removeItems 115, 119, 120
removeTextContent.....283
ReplaceDescriptor.......332
ReplaceString..............329
Resume..........................96
Resume Next.................96
RFC 1738....................131
RGB.....................376, 397
Right........................34, 37
RightBorder.........305, 315
RightBorderDistance...316
RightMargin................315
RmDir............................82
RMZ.............................77
rnd.................................30
RotateAngle.................404
rotieren........................404
Rows....................190, 193
Rtrim........................34, 37

S

Sammlungen................156
Satz..............................282
Schaltflächen...............103
Schaltjahr.......................55
Schattenattribute..........398
scheren.........................404
Schleifen........................61
Schließen.....................177
Schnittstelle..........147148,
Schraffuren..................391

Schriftart......................294
Schriftfarbe..................294
Schriftgröße.................294
Schriftmodus...............294
Schusterjungenreglung 304
screenDown.................288
screenUp......................288
Script...........................364
SDBC..........................418
SearchAll.....................329
SearchDescriptor.........332
SearchRegularExpression329
SearchString................329
Second.....................46, 47
seekGoal......................253
Segment.......................381
Seitenattribute.............312
Seitenformat................243
Seitenhintergrund........243
Seitenrand....244, 312, 315
Seitenübergänge..........409
Seitenzahl....................365
SELECT......................424
Select Case....................42
selectedItem.........115, 119
selectedItemPos...115, 119
SelectedItems……117, 119
SelectedItemsPos.........119
SelectedText................120
selectItem............115, 119
selectItemPos.......115, 119
selectItemsPos.............119
Sequenzielle Dateien.....86
Service................144, 146
SetAttr...........................81
setDropDownLineCount120
setEditable...................120
setEnable.............112, 120
SetExpression..............364
setFocus.......112, 120, 121
setFormula...................201
setLabel.......................112
setMultipleMode.........120
SetPosSize...................122
setPropertyValue.........152
setString.......277, 287, 288
setTitle.........................102
setVisible.....112, 120, 288
sgn................................30
Shadow........................398
ShadowColor...............398
ShadowFormat............316
ShearAngle..................404
Sheets..........................180
Shell..............................88
sin.................................30
Single......................22, 23
SOFT_HYPHEN.........281
Sort.............................250
Sortieren......................250
Sound...........................410
Space.............................37
Spalten.........190, 340, 350
Spaltenbreite................192
Speed...........................410
Speicherdatum...............85
Split........................36, 37
Sprunganweisungen……61
Sprungmarke.................71
Sprungmarken.............278
SQL.............................418
sqr.................................30
StarDesktop.................166
State.............................108
Static..............................25
Step........................61, 123
Stern............................374
Steuerelemente............102
Steuerzeichen..............281
Stil...............................321
storeAsURL................173
Str..................................37
StrComp........................37
String.......................22, 23
Stringverkettung............32
structs..........................159
Strukturen....................159
StyleFamilies...............321
Stylist..........................322
Sub...................................4
SubtitleShape...............377
SUCHEN.......................77
Suchen und Ersetzen328, 408
Suchen_Von_Rechts.….78
supportsService...........164
Surround......................355
Symbol..........................80

T

Tabelle.........................272
Tabellen.......337, 338, 340
TableCount..................364
TableFormula..............364
Tabstopps....................303
Tabulator.............279, 309
tan.................................30
Tastenkombination........80
TemplateName............364
Text.….106, 116, 120, 130
Textattribute................396
TextAutoGrowHeight. 396
TextAutoGrowWidth. .396
Textfeld.......................366
Textfelder….105, 274, 359

TextHorizontalAdjust..396
Textmarken. 278, 337, 351
Textrahmen.........353, 355
TextShape....................377
TextTables...................339
Textverkettung..............32
TextVerticalAdjust......397
Then...............................40
ThisComponent...........145
tieferstellen..................302
Time..............................47
TimeSerial.....................47
TimeValue.....................47
Title.............................102
TitleTextShape............377
TopBorder...........305, 315
TopBorderDistance.....316
TopMargin...................315
Transparency...............389
Trim.........................34, 37
türkischen Buchstaben. 290
Type.............................199

U

Ucase.............................37
Uhrzeitfelder...............130
umbenennen................183
Umlaut.........................277
Umwandlungsfunktionen24
ungroup.......................399
Universal Network Objects
................................144
UNO............................144
UnoFixedTextControl. 133
UnoListBoxControl.....133
unprotect.....................189
Unterprogramme...........53
Until...............................62
UPDATABLE.............427
URL.......................88, 172
User.............................364
Userform.......................99

V

Val.................................37
Validation....................222
Value...................107, 130
Variablen.................17, 19
Variant...........................23
Vektorgrafik................371
verbinden....................404
verbundene Zellen.......346
Vergleiche.....................38
vergrößern...................401
verschieben.................400
Verzweigungen.............40
Vieleck........................387
View............................420
Visible..............................
102, 106, 112, 119, 120
Void............................155
VS.NET..................2, 144

W

Währungsfelder...........130
Währungssymbol.........107
Wait...............................97
Weekday............46, 47, 49
While.............................62
Width...........191, 313, 377
WordCount..................364
Writer..........................272
Wurzel.....................74, 77
WURZEL05..................76
WURZEL2....................76

X

X 377
x-Achse.......................384
Xor................................39

Y

Y 377
y-Achse.......................384
Year.........................46, 47

Z

Zahlenformat...............217
ZÄHLENWENN...........77
Zählerschleife................61
Zeichenblatt.................378
Zeichnung....................371
Zeilen..................190, 340
Zeilen und Spalten einfügen
................................193
Zeilen und Spalten löschen
................................193
Zeilenabstand..............303
Zeilenhöhe...................192
Zeilenwechsel................32
Zellbereich...................195
Zellcursor....................229
Zelle............................195
Zellen..........................194
Zellen verbinden..........220
Zellinhalte löschen......205
Zellschutz....................214
Zielwertsuche..............253